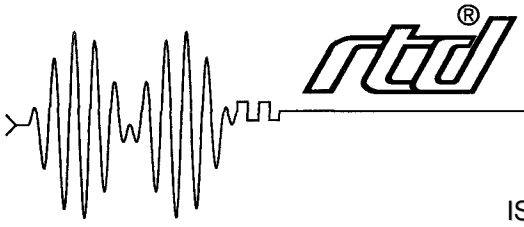
# TC1024
# User's Manual



Real Time Devices, Inc.

*"Accessing the Analog World"®*

ISO9001 and AS9100 Certified

# TC1024

# User's Manual

**REAL TIME DEVICES, INC.**
Post Office Box 906
State College, Pennsylvania 16804
Phone: (814) 234-8087
FAX: (814) 234-5218

Published by
Real Time Devices, Inc.
P.O. Box 906
State College, PA 16804



Printed in U.S.A.

*Rev. B 9411*

# Table of Contents

# List of Illustrations

# INTRODUCTION

The TC1024 Advanced Industrial Control board turns your IBM PC/AT or compatible into a high-performance timing, counting, and control system. Installed within a single expansion slot in the computer, the TC1024 features:

- 10 general purpose 16-bit timer/counters (two Am9513A chips),
- 24 timer/counter modes of operation,
- Binary or BCD up or down counting,
- 16-bit transfers using AT data bus,
- Cascading of up to 10 counters, 160 bits,
- Pads for adding filters on clock input lines,
- On-board 5 MHz oscillator,
- 24 buffered TTL/CMOS 8255-based digital I/O lines with optional pull-up or pull-down resistors,
- 11 hardware configurable interrupts,
- +5 volts only operation,
- Turbo Pascal and Turbo C source code; diagnostics program.

The following paragraphs briefly describe the major functions of the board. A more detailed discussion of board functions is included in Chapter 3, *Hardware Operation*, and Chapter 4, *Board Operation and Programming*. The board setup is described in Chapter 1, *Board Settings*.

## Am9513A Timer/Counter

The versatile Am9513A general purpose timer/counter provides a variety of timing, sequencing, and counting functions. The Am9513A chip contains five 16-bit counters which can be used individually or internally cascaded to form a counter of up to 80 bits. The TC1024 has two Am9513A chips: Am9513A #1 contains counters 1 through 5 and Am9513A #2 contains counters 6 through 10. With 24 operating modes, up or down counting in binary or BCD, and hardware or software gating, these counters can be easily tailored for a wide variety of applications. The counters are clocked by an on-board 5 MHz crystal. On-board RC pads let you custom filter each clock input line for switch debouncing and elimination of unwanted ringing. The source, gate, and output for each counter is available at the P2 I/O connector.

## Digital I/O

The TC1024 has 24 TTL/CMOS-compatible digital I/O lines which can be directly interfaced with external devices or signals to sense switch closures, trigger digital events, or activate solid-state relays. These lines are provided by the on-board 8255 programmable peripheral interface chip. The 8255 can be operated in any one of the three available modes: Mode 0, Mode 1, or Mode 2. To ensure high driving capacity in Mode 0, CMOS buffers are installed. These buffers can be bypassed to support Mode 1 or 2 operation. TTL buffers are available on request.

Pads for installing and activating pull-up or pull-down resistors are included on the board. Installation procedures are given at the end of Chapter 1, *Board Settings*.

## What Comes With Your Board

You receive the following items in your TC1024 package:

- TC1024 AT interface board
- Software and diagnostics diskette with Turbo Pascal and Turbo C source code
- User's manual

If any item is missing or damaged, please call Real Time Devices' Customer Service Department at (814) 234-8087. If you require service outside the U.S., contact your local distributor.

## Board Accessories

In addition to the items included in your TC1024 package, Real Time Devices offers a full line of board accessories. Call your local distributor or our main office for more information about these accessories and for help in choosing the best items to support your board's application. Accessories for the TC1024 include the TB50 terminal board and XB50 prototype/terminal board for prototype development and easy signal access, and XT50 twisted pair wire flat ribbon cable assembly for external interfacing.

## Using This Manual

This manual is intended to help you install your new board and get it running quickly, while also providing enough detail about the board and its functions so that you can enjoy maximum use of its features even in the most complex applications. We assume that you already have an understanding of data acquisition and control principles and that you can customize the example software or write your own applications programs.

## When You Need Help

This manual and the example programs in the software package included with your board provide enough information to properly use all of the board's features. If you have any problems installing or using this board, contact our Technical Support Department, (814) 234-8087, during regular business hours, eastern standard time or eastern daylight time, or send a FAX requesting assistance to (814) 234-5218. When sending a FAX request, please include your company's name and address, your name, your telephone number, and a brief description of the problem.

# CHAPTER 1

## BOARD SETTINGS

The TC1024 has jumper and switch settings you can change if necessary for your application. The board is factory-configured as listed in Table 1-1 and shown on the board layout in the beginning of this chapter. Should you need to change these settings, use these easy-to-follow instructions before you install the board in your computer.

To increase your flexibility in using interrupts, a wire-wrap header is provided at P7 so that you can connect any one of 11 interrupt sources to any one of 11 interrupt channels.

Note that by installing resistor packs at the locations labeled to the right of the 8255 PPI and soldering jumpers as desired on the associated pads, you can configure your 8255 digital I/O lines to be pulled up or pulled down. This procedure is explained near the end of this chapter.

Pads are provided in the upper right area of the board so that you can add custom resistor-capacitor filtering on each clock source input line for switch debouncing and to eliminate unwanted ringing.

## Factory-Configured Switch and Jumper Settings

Table 1-1 lists the factory settings of the user-configurable jumpers and switches on the TC1024 board. Figure 1-1 shows the board layout and the locations of the factory-set jumpers. The following paragraphs explain how to change the factory settings. Pay special attention to the setting of S1, the base address switch, to avoid address contention when you first use your board in your system.

**Table 1-1 — Factory Settings**

| Switch/ Jumper | Function Controlled | Factory Settings (Jumpers Installed) |
|---|---|---|
| P5 | Connects the output of counter 2 or 5 to an interrupt channel (S4 selects which counter is available) | Interrupt channels disabled |
| P6 | Connects the output of counter 7 or 10 to an interrupt channel (S4 selects which counter is available) | Interrupt channels disabled |
| P7 | Connects any of 11 interrupt sources to any of 11 interrupt channels; connection made by wire wrapping between selected header pins | No connection |
| S1 | Sets the base address | 300 hex (768 decimal) |
| S2 | Bypasses 8255 Port C buffers for Mode 1 or Mode 2 operation | Open (buffers not bypassed) |
| S3 | Bypasses 8255 Port A buffers for Mode 2 operation | Open (buffers not bypassed) |
| S4 | Selects the interrupt sources to be available at P5 and P6; selects the source for counter 1 and counter 6 | OUT5, OUT10, EXT6, EXT1 |

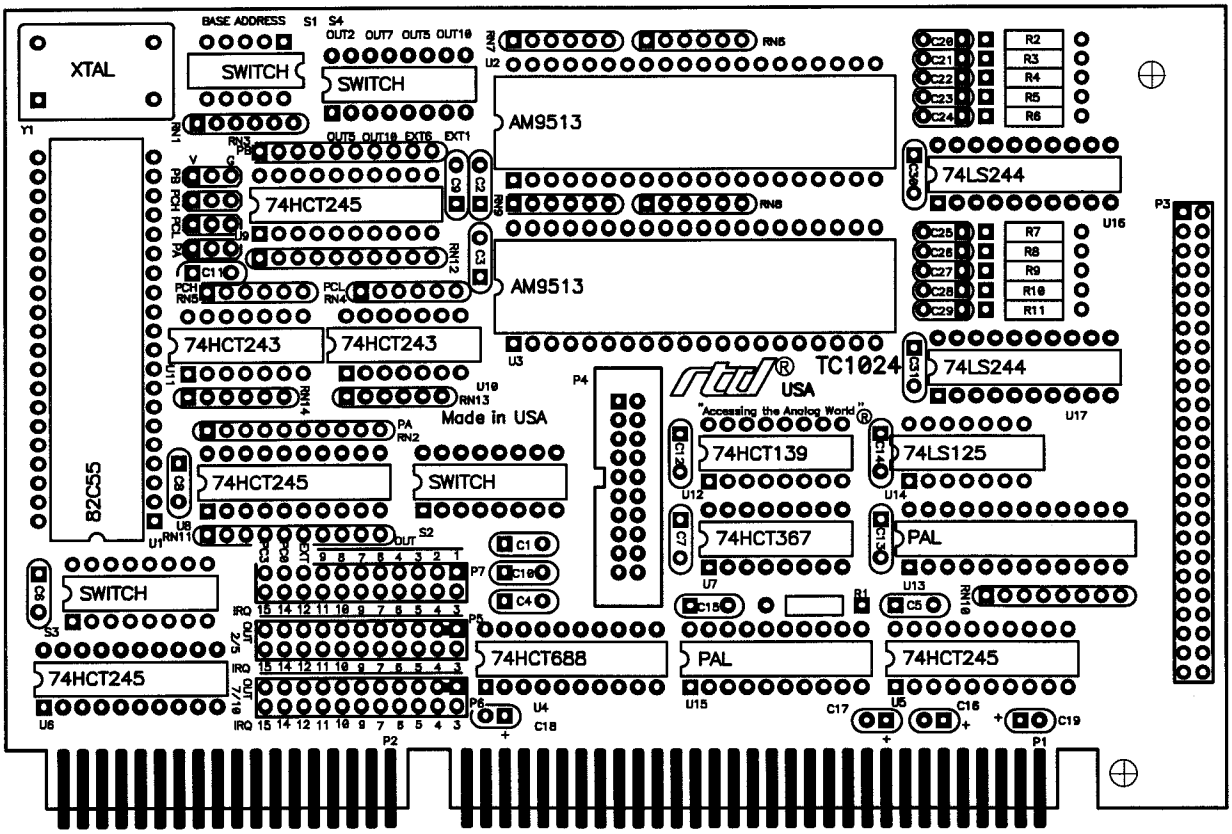



Fig. 1-1 — Board Layout Showing Factory-Configured Settings

**P5 — Counter OUT 2/5 Interrupt Channel Select (Factory Setting: Interrupt Channels Disabled)**

This header connector, shown in Figure 1-2, lets you connect the output of counter 2 or 5, whichever is selected on S4-1, to any of 11 interrupt channels, IRQ9 (highest priority channel) through IRQ12, IRQ14, IRQ15, and then back to IRQ3 through IRQ7 (lowest priority). Chapter 4 explains interrupt channel prioritization in detail. To activate a channel, you must install a jumper across the desired IRQ channel. Figure 1-2a shows the factory setting; Figure 1-2b shows the interrupt source connected to IRQ3. If you use multiple interrupts, make sure each source is assigned to a different IRQ channel.

Fig. 1-2a: Factory Setting

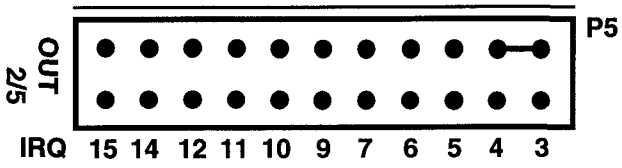



Fig. 1-2b: Interrupt Source Connected to IRQ3

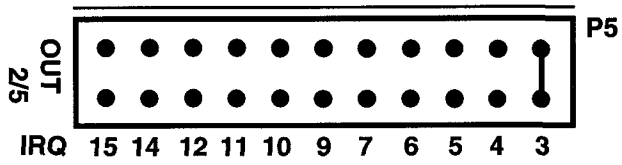



Fig. 1-2 — Counter OUT 2/5 Interrupt Channel Select Jumper, P5

**P6 — Counter OUT 7/10 Interrupt Channel Select (Factory Setting: Interrupt Channels Disabled)**

This header connector, shown in Figure 1-3, lets you connect the output of counter 7 or 10, whichever is selected on S4-2, to any of 11 interrupt channels, IRQ9 (highest priority channel) through IRQ12, IRQ14, IRQ15, and then back to IRQ3 through IRQ7 (lowest priority). Chapter 4 explains interrupt channel prioritization in detail. To activate a channel, you must install a jumper across the desired IRQ channel. Figure 1-3a shows the factory setting; Figure 1-3b shows the interrupt source connected to IRQ9. If you use multiple interrupts, make sure each source is assigned to a different IRQ channel.

Fig. 1-3a: Factory Setting

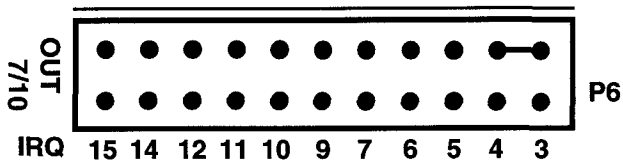



Fig. 1-3b: Interrupt Source Connected to IRQ9

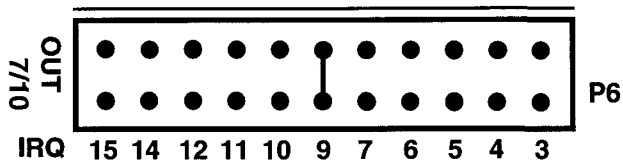



Fig. 1-3 — Counter OUT 7/10 Interrupt Channel Select Jumper, P6

**P7 — Interrupt Source/Channel Select (Factory Setting: No Connection)**

This wire-wrap header connector, shown in Figure 1-4, lets you connect any of 11 interrupt sources to any of the 11 available interrupt channels by wire-wrapping between the appropriate pins on the header. Designed to provide maximum flexibility in interrupt source selection, the interrupt sources provided are: PC3, which is the INTRA signal from the 8255 PPI; PC0, which is the INTRB signal from the 8255 PPI; EXT, an external interrupt you can route onto the board through the P2 I/O connector, and eight of the 10 counter outputs (counters 5 and 10

are not provided; however, they are available at P5 and P6). You can wire-wrap any source to any of 11 interrupt channels, IRQ9 (highest priority channel) through IRQ12, IRQ14, IRQ15, and then back to IRQ3 through IRQ7 (lowest priority). Chapter 4 explains interrupt channel prioritization in detail. If you use multiple interrupts, make sure each source is assigned to a different IRQ channel.

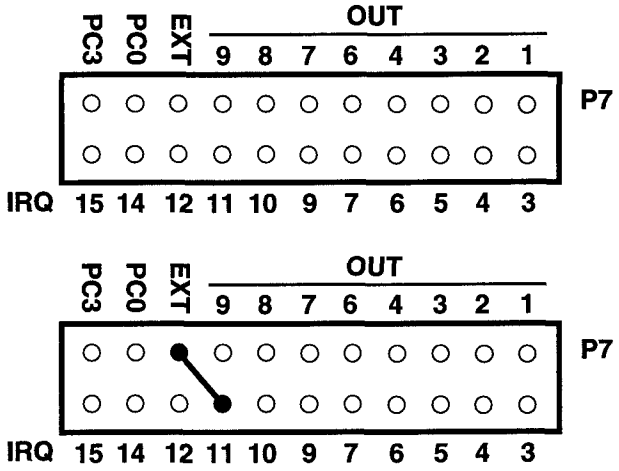



Fig. 1-4a: Factory Setting

Fig. 1-4b: EXT Connected to IRQ11

Fig. 1-4 — Interrupt Source/Channel Select Wire-Wrap Header, P7

**S1 — Base Address (Factory Setting: 300 hex (768 decimal))**

One of the most common causes of failure when you are first trying your board is address contention. Some of your computer's I/O space is already occupied by internal I/O and other peripherals. When the TC1024 board attempts to use I/O address locations already used by another device, contention results and the board does not work.

To avoid this problem, the TC1024 has an easily accessible DIP switch, S1, which lets you select any one of 32 starting addresses in the computer's I/O. Should the factory setting of 300 hex (768 decimal) be unsuitable for your system, you can select a different base address simply by setting the switches to any value shown in Table 1-2. The table shows the switch settings and their corresponding decimal and hexadecimal (in parentheses) values. Make sure that you verify the order of the switch numbers on the switch (1 through 5) before setting them. When the switches are pulled forward, they are OPEN, or set to logic 1, as labeled on the DIP switch package. When you set the base address for your board, record the value in the table inside the back cover. Figure 1-5 shows the DIP switch set for a base address of 300 hex (768 decimal).

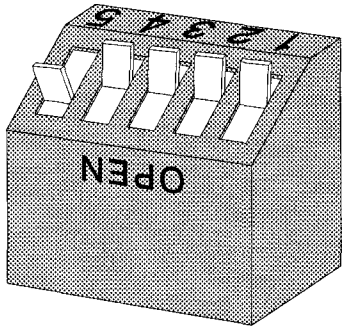



Fig. 1-5 — Base Address Switch, S1

| Table 1-2 — Base Address Switch Settings, S1 | | | |
|---|---|---|---|
| **Base Address Decimal / (Hex)** | **Switch Setting 5 4 3 2 1** | **Base Address Decimal / (Hex)** | **Switch Setting 5 4 3 2 1** |
| 512 / (200) | 0 0 0 0 0 | 768 / (300) | 1 0 0 0 0 |
| 528 / (210) | 0 0 0 0 1 | 784 / (310) | 1 0 0 0 1 |
| 544 / (220) | 0 0 0 1 0 | 800 / (320) | 1 0 0 1 0 |
| 560 / (230) | 0 0 0 1 1 | 816 / (330) | 1 0 0 1 1 |
| 576 / (240) | 0 0 1 0 0 | 832 / (340) | 1 0 1 0 0 |
| 592 / (250) | 0 0 1 0 1 | 848 / (350) | 1 0 1 0 1 |
| 608 / (260) | 0 0 1 1 0 | 864 / (360) | 1 0 1 1 0 |
| 624 / (270) | 0 0 1 1 1 | 880 / (370) | 1 0 1 1 1 |
| 640 / (280) | 0 1 0 0 0 | 896 / (380) | 1 1 0 0 0 |
| 656 / (290) | 0 1 0 0 1 | 912 / (390) | 1 1 0 0 1 |
| 672 / (2A0) | 0 1 0 1 0 | 928 / (3A0) | 1 1 0 1 0 |
| 688 / (2B0) | 0 1 0 1 1 | 944 / (3B0) | 1 1 0 1 1 |
| 704 / (2C0) | 0 1 1 0 0 | 960 / (3C0) | 1 1 1 0 0 |
| 720 / (2D0) | 0 1 1 0 1 | 976 / (3D0) | 1 1 1 0 1 |
| 736 / (2E0) | 0 1 1 1 0 | 992 / (3E0) | 1 1 1 1 0 |
| 752 / (2F0) | 0 1 1 1 1 | 1008 / (3F0) | 1 1 1 1 1 |
| **0 = closed, 1 = open** | | | |

**S2 and S3 — Buffer Bypass Switches (Factory Setting: OPEN (Not Bypassed))**

**Mode 1 Operation (S2)** — When operating the 8255 in Mode 1, the lines of Port C function as control lines, some as outputs and some as inputs. When using Mode 1, the Port C buffers must be removed and bypassed to allow the Port C lines to be individually set as inputs or outputs. Figure 1-6 shows the Port C buffers, and the following steps tell you how to configure the board for Mode 1 operation.

To remove buffering from Port C:

1. Close DIP switches 1 through 8 on S2.
2. Remove U10 from the board.
3. Remove U11 from the board.

**CAUTION:** Remember, whenever you close the switches on S2, **be sure** to remove the Port C buffers, U10 and U11, from the board. Failure to do so may damage the board.

**Mode 2 Operation (S2, S3)** — When operating the 8255 in Mode 2, the lines of Port A must be bidirectional and the lines of Port C function as control lines, some as outputs and some as inputs. When using Mode 2, both the Port A and Port C buffers must be removed and bypassed. Figure 1-7 shows the Port A buffers, Figure 1-6 shows the Port C buffers, and the following steps tell you how to configure the board for Mode 2 operation.

To remove buffering from Ports A and C:

1. Close DIP switches 1 through 8 on S3 (Port A).
2. Remove U8 from the board.
3. Close DIP switches 1 through 8 on S2 (Port C).
4. Remove U10 from the board.
5. Remove U11 from the board.

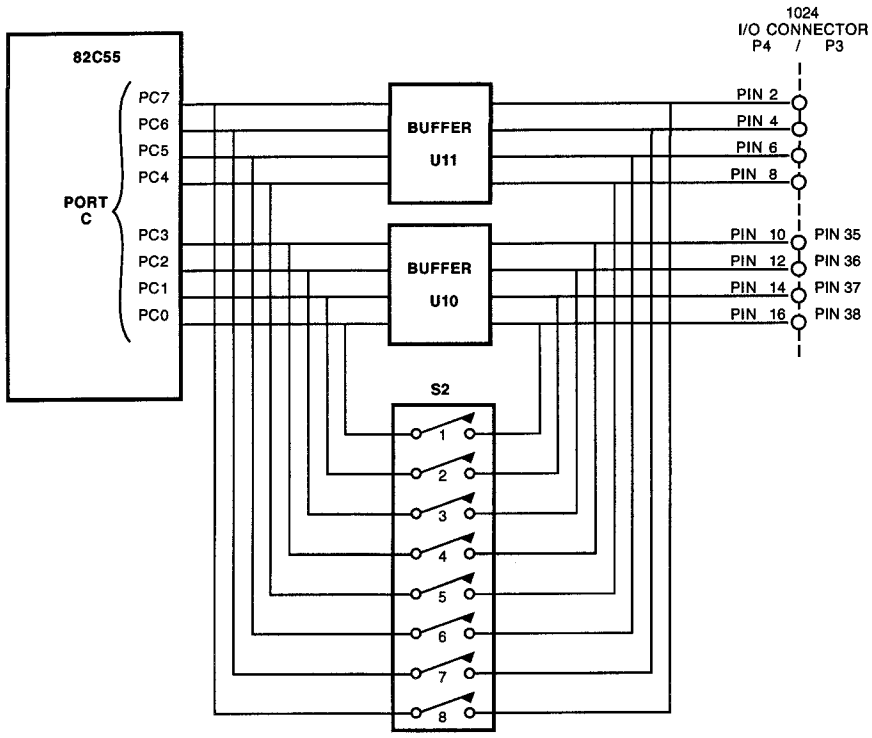



Fig. 1-6 — Port C Buffer Circuitry

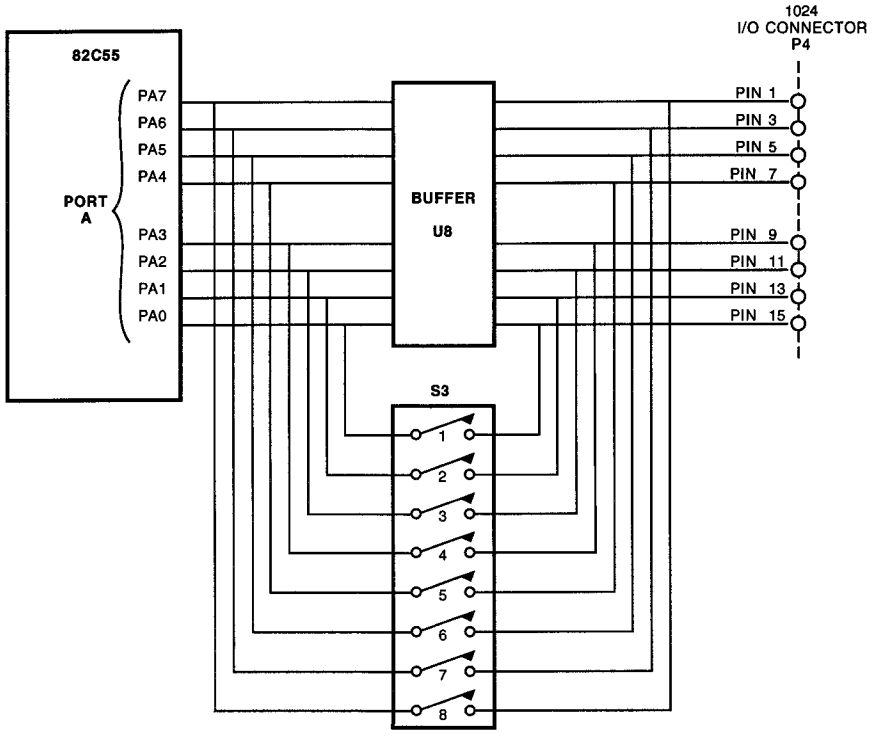



Fig. 1-7 — Port A Buffer Circuitry

**CAUTION:** Remember, whenever you close the switches on S2 and S3, **be sure** to remove the buffers, U8, U10, and U11, from the board. Failure to do so may damage the board.

**S4 — Interrupt Source/Clock Source Select (Factory Setting: OUT5, OUT10, EXT6, EXT1)**

These four single-pole, double-throw switches, shown in Figure 1-8, let you select which counter output provides the interrupt source for P5 and P6, and let you select the clock source for counters 1 and 6.

**S4-1.** This switch provides the output of counter 5 or the output of counter 2 as the available interrupt source at P5. The factory setting is OUT5. Figure 1-9 shows how this switch is connected.

**S4-2.** This switch provides the output of counter 10 or the output of counter 7 as the available interrupt source at P6. The factory setting is OUT10. Figure 1-9 shows how this switch is connected.

**S4-3.** This switch controls the clock source for counter 6. The source can be provided externally from the P3 I/O connector, or it can be provided from the output of counter 5, cascading counter 6 to counter 5. The factory setting is external, EXT6 (SRC 6 at the I/O connector). Figure 1-9 shows how this switch is connected.

**S4-4.** This switch controls the clock source for counter 1. The source can be provided externally from the P3 I/O connector, or it can be provided from the output of counter 10, looping counter 10's output back around to counter 1. The factory setting is external, EXT1 (SRC 1 at the I/O connector). Figure 1-9 shows how this switch is connected.

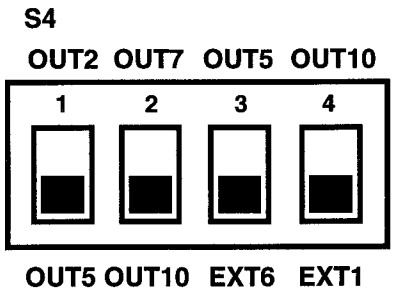



Fig. 1-8 — Interrupt Source/Clock Source Select SPDT Switch, S4

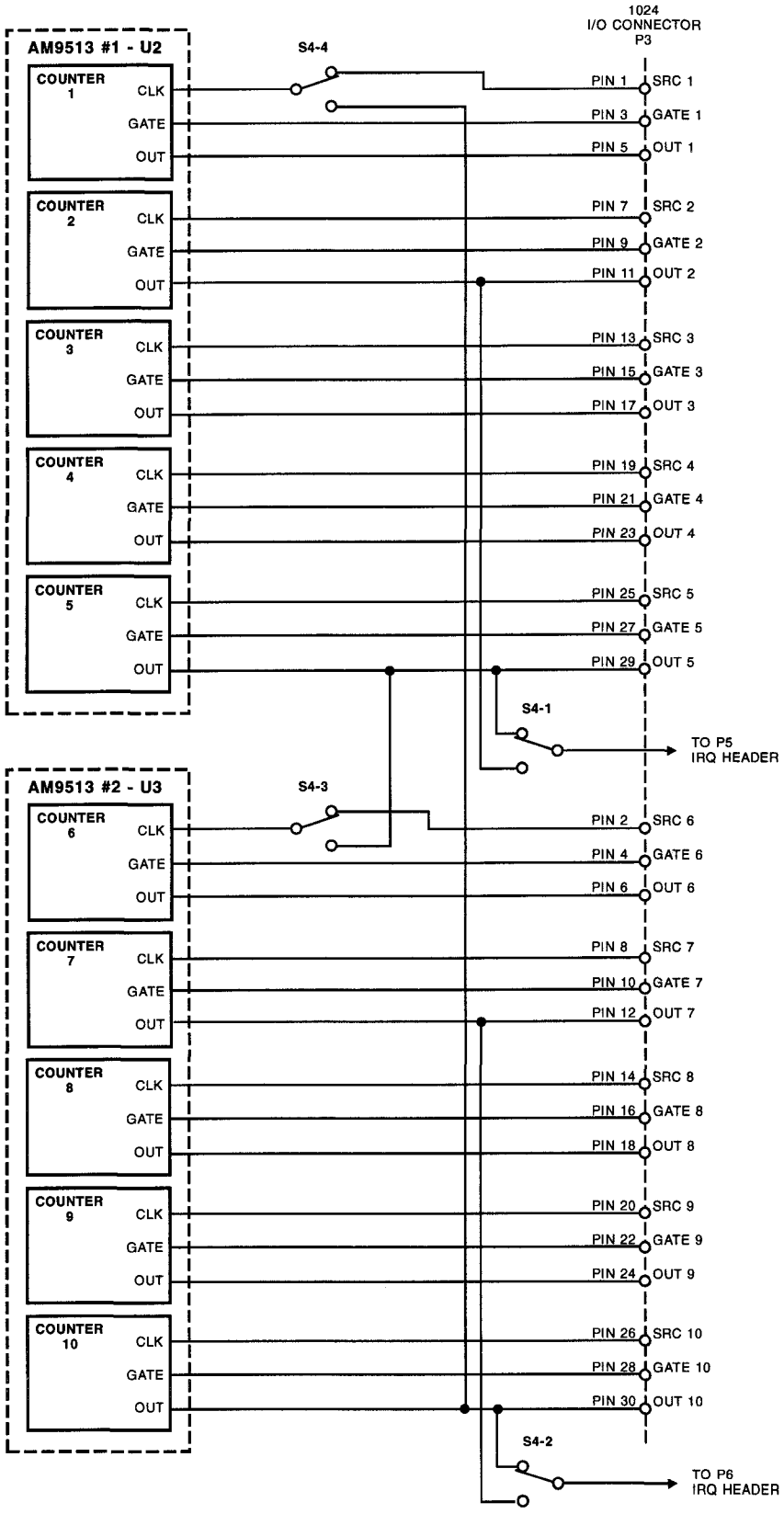



Fig. 1-9 — Counter Circuitry Showing S4 Switch Connections

## Pull-up/Pull-down Resistors on Digital I/O Lines

The 8255 programmable peripheral interface provides 24 TTL/CMOS compatible digital I/O lines which can be interfaced with external devices. The lines are divided into four groups: eight Port A lines, four Port C Lower lines, eight Port B lines, and four Port C Upper lines. You can install and connect pull-up or pull-down resistors for any or all of these four groups of lines. You may want to pull lines up for connection to switches. This will pull the line high when the switch is disconnected. Or, you may want to pull down lines connected to relays which control turning motors on and off. These motors turn on when the digital lines controlling them are high. The Port A and Port B lines of the 8255 automatically power up as inputs – which can float high – during the few moments before the board is initialized. This can cause external devices connected to these lines to operate erratically. By pulling these lines down, when the data acquisition system is first turned on, the motors will not switch on before the 8255 is initialized.

To use the pull-up/pull-down feature, you must first install resistor packs in any or all of the four locations around the 8255, labeled PA, PB, PCL, and PCH. PA and PB take 10-pin packs, and PCL and PCH take 6-pin packs. Figure 1-10 shows a blowup of this circuitry.

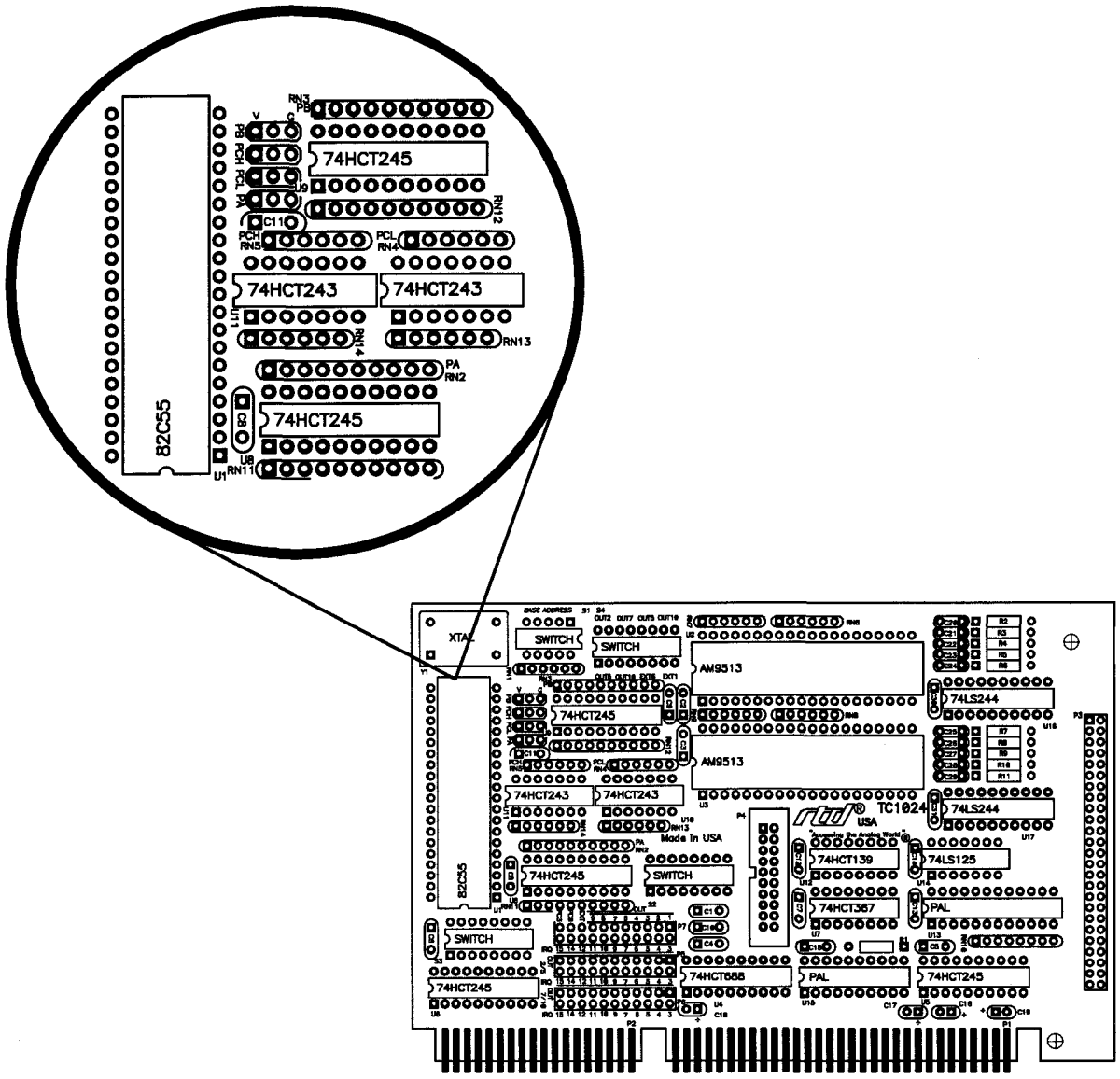



Fig. 1-10 — Pull-up/Pull-down Resistor Circuitry

After the resistor packs are installed, you must connect them into the circuit as pull-ups or pull-downs. Locate the three-hole pads on the board near the resistor packs. They are labeled G (for ground) on one end and V (for +5V) on the other end. The middle hole is common. PA is for Port A, PB for Port B, PCL is for Port C Lower, and PCH is for Port C Upper. Figure 1-10 shows a blowup of the pads. To operate as pull-ups, solder a jumper wire between the common pin (middle pin of the three) and the V pin. For pull-downs, solder a jumper wire between the common pin (middle pin) and the G pin. Figure 1-11 shows Port A lines with pull-ups, Port C Lower with pull-downs, and Port C Upper with no resistors.

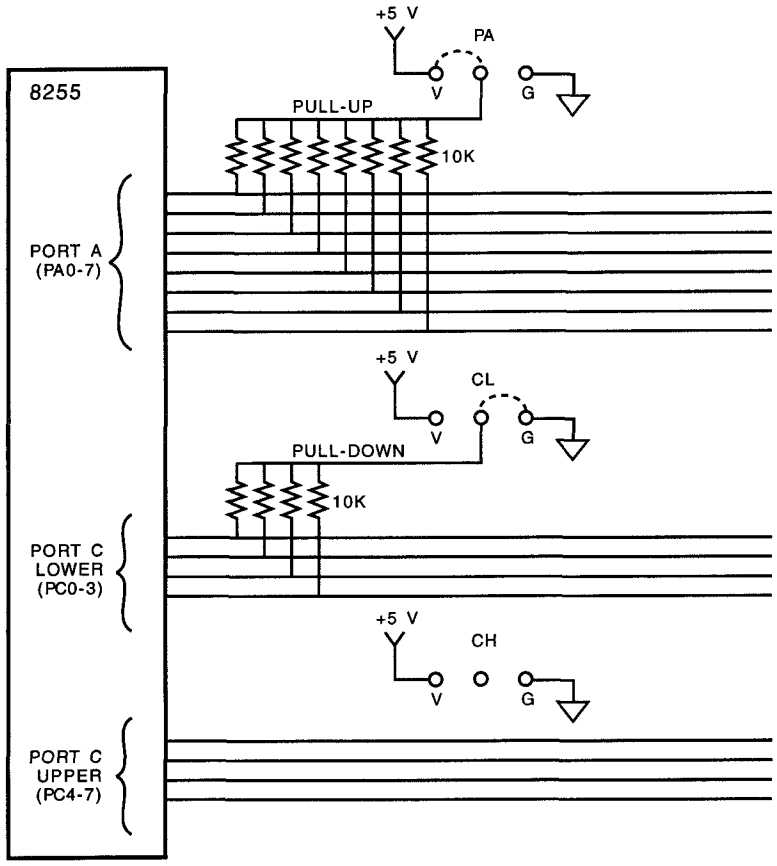



Fig. 1-11 — Adding Pull-ups and Pull-downs to Some Digital I/O Lines

## RC Filters on Source Clock Input Lines

On-board pads are provided to add custom filtering on each of the 10 source clock input lines. These RC pads, located in the upper right area of the board, let you build a switch debouncing circuit or a circuit to eliminate uwanted ringing on the clock input. Simply calculate the values you want to use to achieve the desired results and solder the components onto the board. Figure 1-12 shows a switch debouncing circuit for the TC1024 source clock inputs. Table 1-3 lists the resistor-capacitor pairs for each source clock input line.

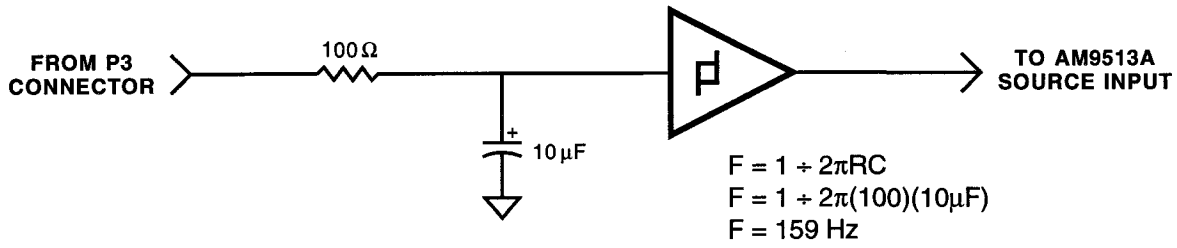



Fig. 1-12 — Typical Switch Debouncing Filter Circuit

**Table 1-3 — Source Clock RC Filters**

| Source Clock Number | Resistor Number | Capacitor Number |
|---|---|---|
| 1 | R2 | C20 |
| 2 | R3 | C21 |
| 3 | R4 | C22 |
| 4 | R5 | C23 |
| 5 | R6 | C24 |
| 6 | R7 | C25 |
| 7 | R8 | C26 |
| 8 | R9 | C27 |
| 9 | R10 | C28 |
| 10 | R11 | C29 |

# CHAPTER 2

## BOARD INSTALLATION

The TC1024 is easy to install in your PC/AT or compatible computer. This chapter tells you step-by-step how to install and connect the board.

After you have installed the board and made all of your connections, you can turn your system on and run the 1024DIAG board diagnostics program included on your example software disk to verify that your board is working.

## Board Installation

Keep the board in its antistatic bag until you are ready to install it in your computer. When removing it from the bag, hold the board at the edges and do not touch the components or connectors.

Before installing the board in your computer, check the jumper and switch settings. Chapter 1 reviews the factory settings and how to change them. If you need to change any settings, refer to the appropriate instructions in Chapter 1. Note that incompatible jumper settings can result in unpredictable board operation and erratic response.

To install the board:

1. Turn OFF the power to your AT computer.
2. Remove the top cover of the computer housing (refer to your owner's manual if you do not already know how to do this).
3. Select any unused expansion slot and remove the slot bracket.
4. Touch the metal housing of the computer to discharge any static buildup and then remove the board from its antistatic bag.
5. If you are using the 20-pin P4 connector for 8255 digital I/O operations, connect the mating connector to it before installing the board in the PC. Note that the P3 I/O connector mounting bracket has an oversized cutout to allow space for running the cable to P4 through the same I/O slot. If you want to run both cables through the same slot, you must make these connections before installing the board.
6. Holding the board by its edges, orient it so that its card edge (bus) connectors line up with the expansion slot connectors in the bottom of the selected expansion slot.
7. After carefully positioning the board in the expansion slot so that the card edge connectors are resting on the computer's bus connectors, gently and evenly press down on the board until it is secured in the slot.

   NOTE: Do not force the board into the slot. If the board does not slide into place, remove it and try again. Wiggling the board or exerting too much pressure can result in damage to the board or to the computer.
8. After the board is installed, secure the slot bracket back into place and put the cover back on your computer. The board is now ready to be connected via the external I/O connector at the rear panel of your computer. Be sure to observe the keying when connecting your external cable to I/O connector P3.

## External I/O Connections

Figure 2-1 shows the TC1024's P3 50-pin I/O connector and P4 on-board 20-pin connector pinouts. Refer to these diagrams as you make your I/O connections.

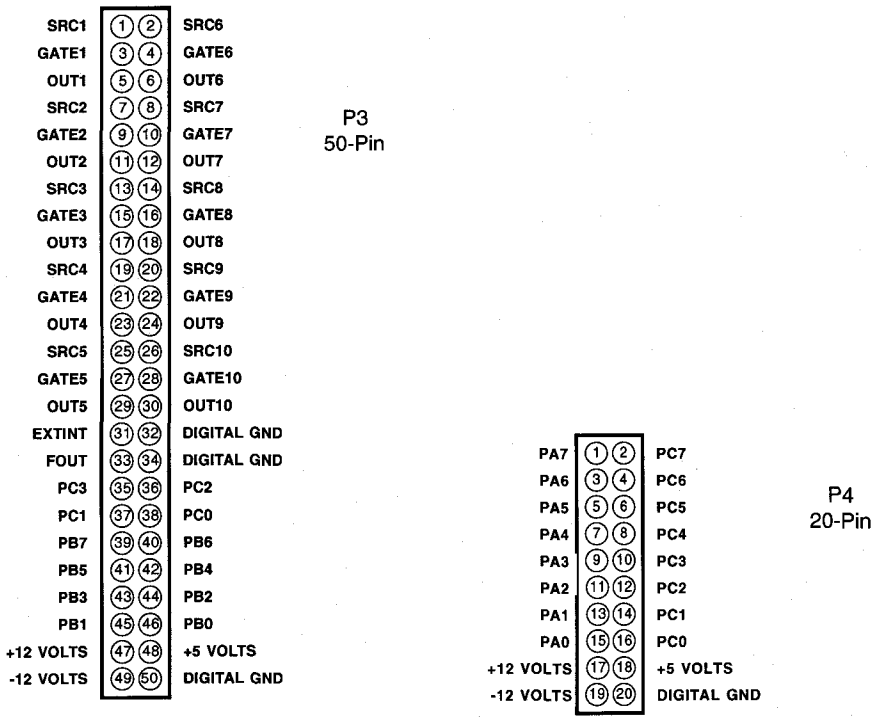



Fig. 2-1 — P3 I/O Connector and P4 On-board Connector Pin Assignments

### Connecting the Timer/Counters and Digital I/O

For all of these connections, the high side of an external signal source or destination device is connected to the appropriate signal pin on the P3 I/O connector or on P4, and the low side is connected to any DIGITAL GND.

## Running the 1024DIAG Diagnostics Program

Now that your board is ready to use, you will want to try it out. An easy-to-use, menu-driven diagnostics program, 1024DIAG, is included with your example software to help you verify your board's operation. You can also use this program to make sure that your current base address setting does not contend with another device.

# CHAPTER 3

## HARDWARE DESCRIPTION

This chapter describes the features of the TC1024 hardware. The major circuits are the timer/counters and the digital I/O lines. This chapter also describes the hardware-selectable interrupts.

The TC1024 board has two major circuits, the timer/counters and the digital I/O lines. Figure 3-1 shows the block diagram of the board. This chapter describes the hardware which makes up the major circuits and hardware-selectable interrupts.

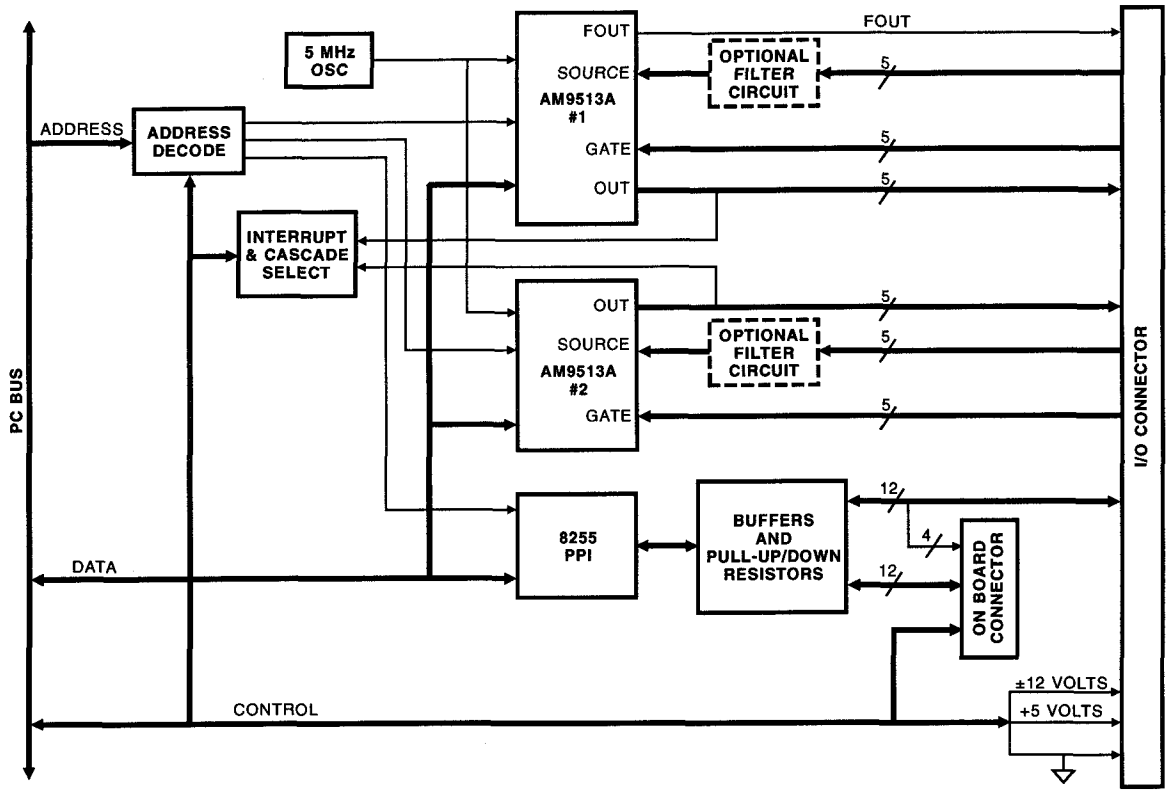



Fig. 3-1 — TC1024 Block Diagram

## Am9513A Timer/Counters

The Am9513A System Timing Controller contains five general purpose 16-bit timer/counters which are capable of performing many different types of counting, sequencing, and timing functions. The Am9513A supports up or down counting in binary or BCD with hardware or software gating of each counter. Its 24 modes of operation are detailed in the Am9513A Data Sheet reprint from AMD included in Appendix C.

The Am9513A is structured with a series of internal registers that set the mode of operation for each counter. These registers are fully described in Appendix C.

Any combination of the 10 counters in the two Am9513As can be internally cascaded to create a counter of up to 160 bits. For example, two cascaded counters form a 32-bit counter for longer counting capability. Rarely is it practical to cascade more than three counters. Cascading is described in Appendix C, Chapter 3 of the Am9513A data sheet.

The timer/counters are driven by an on-board 5 MHz crystal oscillator. On-board RC pads let you custom filter each source clock input line for switch debouncing and elimination of unwanted ringing.

## Digital I/O, Programmable Peripheral Interface

The 8255 programmable peripheral interface (PPI) can be easily configured to solve a wide range of digital real-world problems. This high-performance TTL/CMOS compatible chip has 24 parallel programmable digital I/O lines divided into two groups of 12 lines each:

Group A — Port A (8 lines) and Port C Upper (4 lines);
Group B — Port B (8 lines) and Port C Lower (4 lines).

Each group can be programmed for one of three modes of operation. When operating in Mode 1, the on-board buffers must be removed from the Port C lines. When operating in Mode 2, both Port A and Port C buffering must be removed. This procedure is described in Chapter 1 in the S2 and S3 DIP switch discussion. The three operating modes are:

Mode 0 — Basic input/output. Lets you use simple input and output operation for a port. Data is written to or read from the specified port.

Mode 1 — Strobed input/output. Lets you transfer I/O data from Port A in conjunction with strobes or handshaking signals.

Mode 2 — Strobed bidirectional input/output. Lets you communicate bidirectionally with an external device through Port A. Handshaking is similar to Mode 1.

These modes are detailed in the 8255 Data Sheet, reprinted from Intel in Appendix C.

The bidirectional buffers on the 8255's I/O lines monitor the 8255 control word to automatically set their direction. Hardware changes to the buffer circuitry are required only when using Mode 1 or Mode 2, where the Port A and/or Port C buffers must be removed as described in Chapter 1.

## Interrupts

The TC1024 has several hardware selectable interrupt sources. These interrupt sources can be selected using jumpers or wire-wrapping on P5 through P7, as described in Chapter 1. Interrupt sources which can be used by the TC1024 are: the outputs from all 10 Am9513A counters; PC3, which is the INTRA signal from the 8255 PPI; PC0, which is the INTRB signal from the 8255 PPI; and EXT, an external interrupt you can route onto the board through I/O connector P3. Chapter 1 tells you how to set the jumpers or configure the wire-wrapping on the interrupt header connectors, P5, P6, and P7, and Chapter 4 describes how to program interrupts.

# CHAPTER 4

## BOARD OPERATION AND PROGRAMMING

This chapter shows you how to program your TC1024 board by writing to and reading from the AT bus in 8- or 16-bit words. It provides a complete description of the I/O map and a description of programming operations to aid you in programming. The example programs included on the disk in your board package are listed at the end of this chapter. These programs, written in Turbo C and Turbo Pascal, include source code to simplify your applications programming. Chapter 5 contains examples for setting up the Am9513A's 16-bit counters for specific applications.

## Defining the I/O Map

The I/O map for the TC1024 is shown in Table 4-1 below. As shown, the board occupies 16 consecutive I/O port locations.

Because of the 16-bit structure of the AT bus, every other address location is used, even for 8-bit transfers. Our programming structure uses the 16-bit command to set up and run the Am9513A. All 8255 read/write operations are 8-bit operations.

The base address (designated as BA) can be selected using DIP switch S1 as described in Chapter 1, *Board Settings*. This switch can be accessed without removing the board from the computer. S1 is factory set at 300 hex (768 decimal). The following sections describe the register contents of each address used in the I/O map.

**Table 4-1 — TC1024 I/O Map**

| Register Description | Read Function | Write Function | Address * (Decimal) |
|---|---|---|---|
| 8255 PPI Port A | Read Port A digital input lines | Program Port A digital output lines | BA + 0 |
| 8255 PPI Port B | Read Port B digital input lines | Program Port B digital output lines | BA + 2 |
| 8255 PPI Port C | Read Port C digital input lines | Program Port C digital output lines | BA + 4 |
| 8255 PPI Control Word | Reserved | Program PPI configuration | BA + 6 |
| Am9513A #1 Data Word | Read data register for Counters 1-5 | Program data register for Counters 1-5 | BA + 8 |
| Am9513A #1 Control Word | Read control register for Counters 1-5 | Program control register for Counters 1-5 | BA + 10 |
| Am9513A #2 Data Word | Read data register for Counters 6-10 | Program data register for Counters 6-10 | BA + 12 |
| Am9513A #2 Control Word | Read control register for Counters 6-10 | Program control register for Counters 6-10 | BA + 14 |
| * BA = Base Address | | | |

### BA + 0: PPI Port A — Digital I/O (Read/Write)

**8-bit operation.** Transfers the 8-bit Port A digital input and digital output data between the board and an external device. A read transfers data from the external device, through on-board connector P4, and into PPI Port A; a write transfers the written data from Port A through P4 to an external device.

### BA + 2: PPI Port B — Digital I/O (Read/Write)

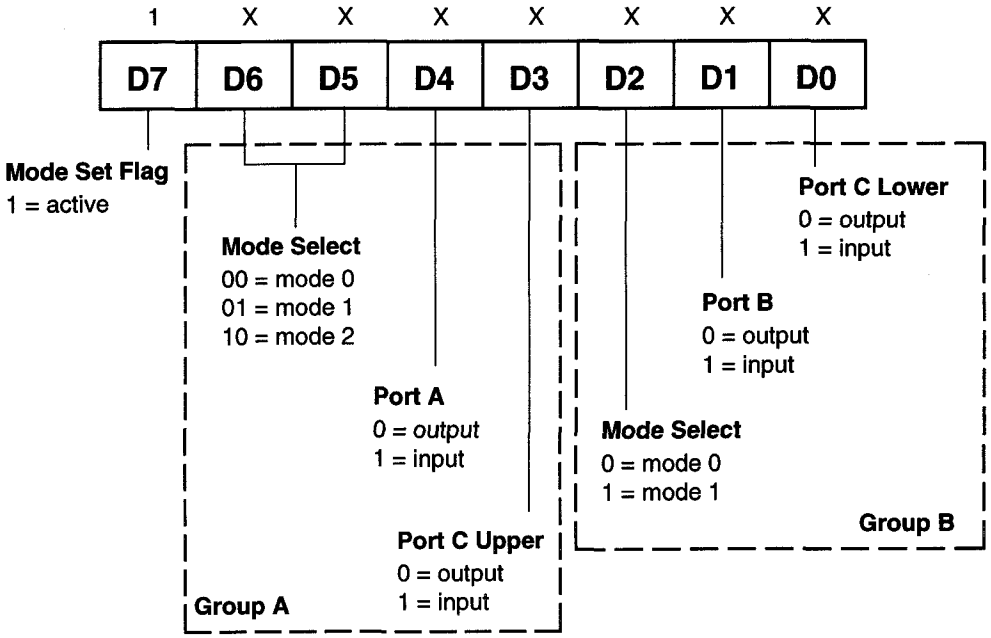
**8-bit operation.** Transfers the 8-bit Port B digital input and digital output data between the board and an external device. A read transfers data from the external device, through external I/O connector P3, and into PPI Port B; a write transfers the written data from Port B through P3 to an external device.

### BA + 4: PPI Port C — Digital I/O (Read/Write)

**8-bit operation.** Transfers the two 4-bit Port C digital input and digital output data groups (Port C Upper and Port C Lower) between the board and an external device. A read transfers data from the external device, through on-board connector P4, and into PPI Port C; a write transfers the written data from Port C through P4 to an external device. The bottom four bits, PC0-PC3, are also brought out to external I/O connector P3.

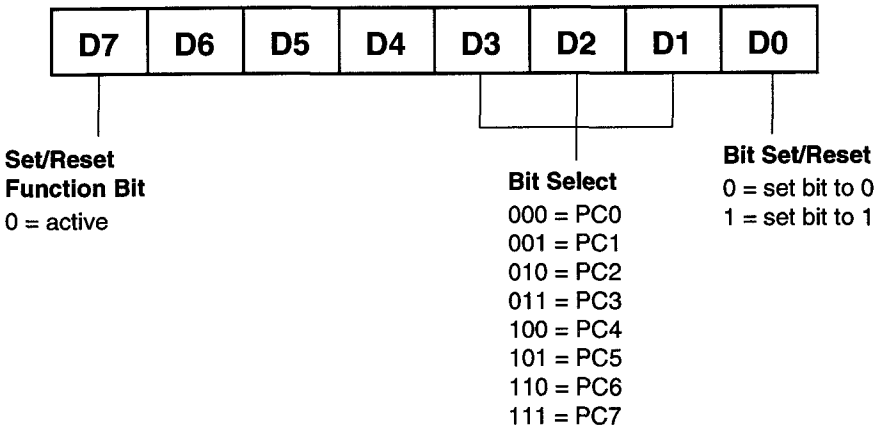
### BA + 6: 8255 PPI Control Word (Write Only)

**8-bit operation.** When bit 7 of this word is set to 1, a write programs the PPI configuration. The table below shows the control words for the 16 possible Mode 0 Port I/O combinations.

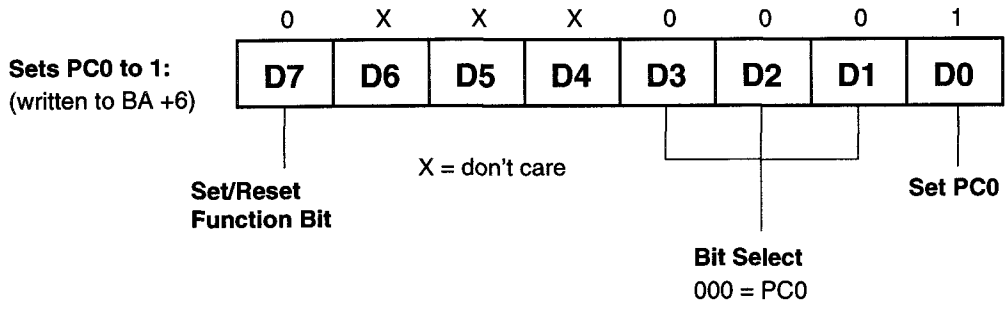



| 8255 Port I/O Flow Direction and Control Words, Mode 0 | | | | | | |
|---|---|---|---|---|---|---|
| Group A | | Group B | | Control Word | | |
| Port A | Port C Upper | Port B | Port C Lower | Binary | Decimal | Hex |
| Output | Output | Output | Output | 10000000 | 128 | 80 |
| Output | Output | Output | Input | 10000001 | 129 | 81 |
| Output | Output | Input | Output | 10000010 | 130 | 82 |
| Output | Output | Input | Input | 10000011 | 131 | 83 |
| Output | Input | Output | Output | 10001000 | 136 | 88 |
| Output | Input | Output | Input | 10001001 | 137 | 89 |
| Output | Input | Input | Output | 10001010 | 138 | 8A |
| Output | Input | Input | Input | 10001011 | 139 | 8B |
| Input | Output | Output | Output | 10010000 | 144 | 90 |
| Input | Output | Output | Input | 10010001 | 145 | 91 |
| Input | Output | Input | Output | 10010010 | 146 | 92 |
| Input | Output | Input | Input | 10010011 | 147 | 93 |
| Input | Input | Output | Output | 10011000 | 152 | 98 |
| Input | Input | Output | Input | 10011001 | 153 | 99 |
| Input | Input | Input | Output | 10011010 | 154 | 9A |
| Input | Input | Input | Input | 10011011 | 155 | 9B |

When bit 7 of this word is set to 0, a write can be used to individually program the Port C lines.




For example, if you want to set Port C bit 0 to 1, you would set up the control word so that bit 7 is 0; bits 1, 2, and 3 are 0 (this selects PC0); and bit 0 is 1 (this sets PC0 to 1). The control word is set up like this:




## IMPORTANT

Because of the bus release time of the Am9513A, AMD recommends you insert a small delay between software accesses to the chip.

**BA + 8: Am9513A #1 Data Register (Read/Write)**

**16-bit operation (after initialization).** Accesses the Am9513A data register for counters 1-5. This chapter explains initialization procedures. See the example programs in Chapter 5 and the data sheet included in Appendix C for more information on the operation of the Am9513A.

**BA + 10: Am9513A #1 Command Register (Read/Write)**

**16-bit operation (after initialization).** Accesses the Am9513A command register for counters 1-5. This chapter explains initialization procedures. See the example programs in Chapter 5 and the data sheet included in Appendix C for more information on the operation of the Am9513A.

**BA + 12: Am9513A #2 Data Register (Read/Write)**

**16-bit operation (after initialization).** Accesses the Am9513A data register for counters 6-10. This chapter explains initialization procedures. See the example programs in Chapter 5 and the data sheet included in Appendix C for more information on the operation of the Am9513A.

**BA + 14: Am9513A #2 Command Register (Read/Write)**

**16-bit operation (after initialization).** Accesses the Am9513A command register for counters 6-10. This chapter explains initialization procedures. See the example programs in Chapter 5 and the data sheet included in Appendix C for more information on the operation of the Am9513A.

## Programming the TC1024

This section gives you some general information about programming and the TC1024 board. Chapter 5 provides some specific programming examples, and the Am9513A data sheet in Appendix C provides detailed programming information for all 24 operating modes of the Am9513A. These tools will help you as you use the example programs included with the board. All of the program descriptions in this section use decimal values unless otherwise specified.

The TC1024 is programmed by writing to and reading from the correct I/O port locations on the board. These I/O ports were defined in the previous section. Because the TC1024 is AT bus compatible, reading/writing the Am9513As is done in a 16-bit word format. All other operations are done in an 8-bit word format. High-level languages such as Pascal, C, and C++ make it very easy to read/write these ports. The table below shows you how to read from and write to I/O ports in Turbo C and Turbo Pascal.

| Language | Read 8 Bits | Write 8 Bits | Read 16 Bits | Write 16 Bits |
|---|---|---|---|---|
| Turbo C | Data = inportb(Address) | outportb(Address, Data) | Data = inport(Address) | outport(Address, Data) |
| Turbo Pascal | Data := Port[Address] | Port[Address] := Data | Data := PortW[Address] | PortW[Address] := Data |

In addition to being able to read/write the I/O ports on the TC1024, you must be able to perform a variety of operations that you might not normally use in your programming. The table below shows you some of the operators discussed in this section, with an example of how each is used with Pascal and C.

| Language | Modulus | Integer Division | AND | OR |
|---|---|---|---|---|
| C | %<br>a = b % c | /<br>a = b / c | &<br>a = b & c | \|<br>a = b \| c |
| Pascal | MOD<br>a := b MOD c | DIV<br>a := b DIV c | AND<br>a := b AND c | OR<br>a := b OR c |

Many compilers have functions that can read/write either 8 or 16 bits from/to an I/O port. For example, Turbo Pascal uses **Port** for 8-bit port operations and **PortW** for 16 bits, Turbo C uses **inportb** for an 8-bit read of a port and **inport** for a 16-bit read. **Be sure to use the correct function for 8-bit and 16-bit operations with the TC1024!**

### Clearing and Setting Bits in a Port

When you clear or set one or more bits in a port, you must be careful that you do not change the status of the other bits. You can preserve the status of all bits you do not wish to change by proper use of the AND and OR binary operators. Using AND and OR, single or multiple bits can be easily cleared in one operation.

To **clear** a single bit in a port, AND the current value of the port with the value b, where $b = 255 - 2^{bit}$.

**Example:** Clear bit 5 in a port. Read in the current value of the port, AND it with 223 ($223 = 255 - 2^5$), and then write the resulting value to the port. In BASIC, this is programmed as:

```
V = INP(PortAddress)
V = V AND 223
OUT PortAddress, V
```

To **set** a single bit in a port, OR the current value of the port with the value b, where $b = 2^{bit}$.

**Example:** Set bit 3 in a port. Read in the current value of the port, OR it with 8 ($8 = 2^3$), and then write the resulting value to the port. In Pascal, this is programmed as:

```
V := Port[PortAddress];
V := V OR 8;
Port[PortAddress] := V;
```

Setting or clearing more than one bit at a time is accomplished just as easily. To **clear** multiple bits in a port, AND the current value of the port with the value b, where b = 255 - (the sum of the values of the bits to be cleared). Note that the bits do not have to be consecutive.

**Example:** Clear bits 2, 4, and 6 in a port. Read in the current value of the port, AND it with 171 ($171 = 255 - 2^2 - 2^4 - 2^6$), and then write the resulting value to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 171;
outportb(port_address, v);
```

To **set** multiple bits in a port, OR the current value of the port with the value b, where b = the sum of the individual bits to be set. Note that the bits to be set do not have to be consecutive.

**Example:** Set bits 3, 5, and 7 in a port. Read in the current value of the port, OR it with 168 ($168 = 2^3 + 2^5 + 2^7$), and then write the resulting value back to the port. In assembly language, this is programmed as:

```
mov dx, PortAddress
in al, dx
or al, 168
out dx, al
```

Often, assigning a range of bits is a mixture of setting and clearing operations. You can set or clear each bit individually or use a faster method of first clearing all the bits in the range then setting only those bits that must be set using the method shown above for setting multiple bits in a port. The following example shows how this two-step operation is done.

**Example:** Assign bits 3, 4, and 5 in a port to 101 (bits 3 and 5 set, bit 4 cleared). First, read in the port and clear bits 3, 4, and 5 by ANDing them with 199. Then set bits 3 and 5 by ORing them with 40, and finally write the resulting value back to the port. In C, this is programmed as:

```
v = inportb(port_address);
v = v & 199;
v = v | 40;
outportb(port_address, v);
```

**A final note:** Don't be intimidated by the binary operators AND and OR and try to use operators for which you have a better intuition. For instance, if you are tempted to use addition and subtraction to set and clear bits in place of the methods shown above, DON'T! Addition and subtraction may seem logical, but they **will not work** if you try to clear a bit that is already clear or set a bit that is already set. For example, you might think that to set bit 5 of a port, you simply need to read in the port, add 32 ($2^5$) to that value, and then write the resulting value back to the port. This works fine if bit 5 is not already set. But, what happens when bit 5 *is* already set? Bits 0 to 4 will be unaffected and we can't say for sure what happens to bits 6 and 7, but we can say for sure that bit 5 ends up cleared instead of being set. A similar problem happens when you use subtraction to clear a bit in place of the method shown above.

Now that you know how to clear and set bits, we are ready to look at the programming steps for the TC1024 board functions.

### Initializing the Am9513A

The Am9513A has a sophisticated internal architecture which is programmed through a series of internal registers. These internal registers are accessed by writing to and reading from only two I/O port locations for each Am9513A on the TC1024 board. For Am9513A #1 which contains counters 1-5, the Data Register port is at BA + 8 and the Control Register port is at BA + 10. For Am9513A #2 which contains counters 6-10, the Data Register port is at BA + 12 and the Control Register port is at BA + 14. In our example programs, follow these steps to initialize the Am9513A. Note that until you point to and set up the master mode register in step 2, you must send your commands in 8-bit format. After the master mode register has been configured for 16-bit operation by setting bit 13 to a logic 1, you can then send 16-bit words to the Am9513A.

1. Send a master reset to the Am9513A (8-bit)
2. Point to and set up the master mode register (8-bit)
   (When setting up the master mode register, set MM13 to logic 1 so that you can do 16-bit transfers)
3. Point to and set up counter 1 mode register (16-bit)
4. Point to counter 1 load register and load desired value (16-bit)
5. Point to and set up counter 2 mode register (16-bit)
6. Point to counter 2 load register and load desired value (16-bit)

•
•

11. Point to and set up counter 5 mode register (16-bit)
12. Point to counter 5 load register and load desired value (16-bit)
13. Load and arm counters (16-bit)

The examples on the disk and in Chapter 5 will aid you in programming the Am9513A for your application. These tools and the data sheet in Appendix C provide a comprehensive description of timer/counter operation.

**IMPORTANT**

Because of the bus release time of the Am9513A, AMD recommends you insert a small delay between software accesses to the chip.

### Initializing the 8255

Before you can use the 24 digital I/O lines on your TC1024, the 8255 PPI must be initialized. This step must be executed every time you start up, reset, or reboot your computer.

The 8255 is initialized by writing the appropriate control word to I/O port BA + 6. The contents of your control word will vary, depending on how you want to configure your I/O lines. Use the control word description in the previous I/O map section to help you program the right value. In the example below, a decimal value of 128 sets up the 8255 so that all I/O lines are Mode 0 outputs. Remember that if you want to use Mode 1 or Mode 2 operation, you must remove the Port A and/or Port C buffers from the board and close the buffer bypass switches. Chapter 1 explains how to do this in the paragraphs covering S2 and S3.

| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
|---|---|---|---|---|---|---|---|
| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |

### Digital I/O Operations

Once the 8255 is initialized, you can use the digital I/O lines to control or monitor external devices.

## Interrupts

### • What Is an Interrupt?

An interrupt is an event that causes the processor in your computer to temporarily halt its current process and execute another routine. Upon completion of the new routine, control is returned to the original routine at the point where its execution was interrupted.

Interrupts are very handy for dealing with asynchronous events (events that occur at less than regular intervals). Keyboard activity is a good example; your computer cannot predict when you might press a key and it would be a waste of processor time for it to do nothing while waiting for a keystroke to occur. Thus, the interrupt scheme is used and the processor proceeds with other tasks. Then, when a keystroke does occur, the keyboard 'interrupts' the processor, and the processor gets the keyboard data, places it in memory, and then returns to what it was doing before it was interrupted. Other common devices that use interrupts are modems, disk drives, and mice.

Your TC1024 board can interrupt the processor when a variety of conditions are met, such as when any of the 10 timer countdowns is finished. Interrupts can also be generated by the 8255 PPI or an external source. By using these interrupts, you can write software that effectively deals with real world events.

### • Interrupt Request Lines

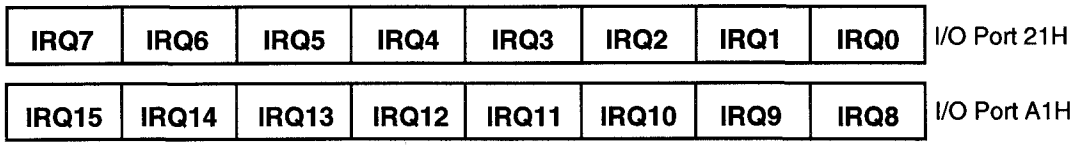
To allow different peripheral devices to generate interrupts on the same computer, the AT bus has 16 different interrupt request (IRQ) lines. A transition from low to high on one of these lines generates an interrupt request which is handled by one of the AT's two interrupt control chips. One chip handles IRQ0 through IRQ7 and the other chip handles IRQ8 through IRQ15. The controller which handles IRQ8-IRQ15 is chained to the first controller through the IRQ2 line. When an IRQ line is brought high, the interrupt controllers check to see if interrupts are to be acknowledged from that IRQ and, if another interrupt is already in progress, they decide if the new request should supersede the one in progress or if it has to wait until the one in progress is done. This prioritizing allows an interrupt to be interrupted if the second request has a higher priority. The priority level is determined by the number of the IRQ. Because of the configuration of the two controllers, with one chained to the other through IRQ2, the priority scheme is a little unusual. IRQ0 has the highest priority, IRQ1 is second-highest, then priority jumps to IRQ8, IRQ9, IRQ10, IRQ11, IRQ12, IRQ13, IRQ14, and IRQ15, and then following IRQ15, it jumps back to IRQ3, IRQ4, IRQ5, IRQ6, and finally, the lowest priority, IRQ7. This sequence makes sense if you consider that the controller that handles IRQ8-IRQ15 is routed through IRQ2.

### • 8259 Programmable Interrupt Controllers

The chips responsible for handling interrupt requests in the PC are the 8259 Programmable Interrupt Controllers. The 8259 that handles IRQ0-IRQ7 is referred to as 8259A, and the 8259 that handles IRQ8-IRQ15 is referred to as 8259B. To use interrupts, you need to know how to read and set the 8259 interrupt mask registers (IMR) and how to send the end-of-interrupt (EOI) command to the 8259s.

### • Interrupt Mask Registers (IMR)

Each bit in the interrupt mask register (IMR) contains the mask status of an IRQ line; in 8259A, bit 0 is for IRQ0, bit 1 is for IRQ1, and so on, while in 8259B, bit 0 is for IRQ8, bit 1 is for IRQ9, and so on. If a bit is **set** (equal to 1), then the corresponding IRQ is masked and it will not generate an interrupt. If a bit is **clear** (equal to 0), then the corresponding IRQ is unmasked and can generate interrupts. The IMR for IRQ0-IRQ7 is programmed through port 21H, and the IMR for IRQ8-IRQ15 is programmed through port A1H.




**For all bits:**
0 = IRQ unmasked (enabled)
1 = IRQ masked (disabled)

**• End-of-Interrupt (EOI) Command**

After an interrupt service routine is complete, the appropriate 8259 interrupt controller must be notified. When using IRQ0-IRQ7, this is done by writing the value 20H to I/O port 20H only; when using IRQ8-IRQ15, you must write the value 20H to I/O ports 20H and A0H.

**• What Exactly Happens When an Interrupt Occurs?**

Understanding the sequence of events when an interrupt is triggered is necessary to properly write software interrupt handlers. When an interrupt request line is driven high by a peripheral device (such as the TC1024), the interrupt controllers check to see if interrupts are enabled for that IRQ, and then check to see if other interrupts are active or requested and determine which interrupt has priority. The interrupt controllers then interrupt the processor. The current code segment (CS), instruction pointer (IP), and flags are pushed on the stack for storage, and a new CS and IP are loaded from a table that exists in the lowest 1024 bytes of memory. This table is referred to as the interrupt vector table and each entry is called an interrupt vector. Once the new CS and IP are loaded from the interrupt vector table, the processor begins executing the code located at CS:IP. When the interrupt routine is completed, the CS, IP, and flags that were pushed on the stack when the interrupt occurred are now popped from the stack and execution resumes from the point where it was interrupted.

**• Using Interrupts in Your Programs**

Adding interrupts to your software is not as difficult as it may seem, and what they add in terms of performance is often worth the effort. Note, however, that although it is not that hard to use interrupts, the smallest mistake will often lead to a system hang that requires a reboot. This can be both frustrating and time-consuming. But, after a few tries, you'll get the bugs worked out and enjoy the benefits of properly executed interrupts. In addition to reading the following paragraphs, study the INTRPTS source code included on your TC1024 program disk for a better understanding of interrupt program development.

**• Writing an Interrupt Service Routine (ISR)**

The first step in adding interrupts to your software is to write the interrupt service routine (ISR). This is the routine that will automatically be executed each time an interrupt request occurs on the specified IRQ. An ISR is different than standard routines that you write. First, on entrance, the processor registers should be pushed onto the stack **BEFORE** you do anything else. Second, just before exiting your ISR, you must write an end-of-interrupt command to the 8259 controller(s). Since 8259B generates a request on IRQ2 which is handled by 8259A, an EOI must be sent to both 8259A and 8259B for IRQ8-IRQ15. Finally, when exiting the ISR, in addition to popping all the registers you pushed on entrance, you must use the IRET instruction and **not** a plain RET. The IRET automatically pops the flags, CS, and IP that were pushed when the interrupt was called.

If you find yourself intimidated by these requirements, take heart. Most Pascal and C compilers allow you to identify a procedure (function) as an interrupt type and will automatically add these instructions to your ISR, with one important exception: most compilers **do not** automatically add the end-of-interrupt command to the procedure; you must do this yourself. Other than this and the few exceptions discussed below, you can write your ISR just like any other routine. It can call other functions and procedures in your program and it can access global data. If you are writing your first ISR, we recommend that you stick to the basics; just something that will convince you that it works, such as incrementing a global variable.

**NOTE:** If you are writing an ISR using assembly language, you are responsible for pushing and popping registers and using IRET instead of RET.

There are a few cautions you must consider when writing your ISR. The most important is, **do not use any DOS functions or routines that call DOS functions from within an ISR**. DOS is **not** reentrant; that is, a DOS function cannot call itself. In typical programming, this will not happen because of the way DOS is written. But what about when using interrupts? Then, you could have a situation such as this in your program. If DOS function X is being executed when an interrupt occurs and the interrupt routine makes a call to DOS function X, then function X is essentially being called while it is already active. Such a reentrancy attempt spells disaster because DOS functions are not written to support it. This is a complex concept and you do not need to understand it. Just make sure that you do not call any DOS functions from within your ISR. The one wrinkle is that, unfortunately, it is not obvious which library routines included with your compiler use DOS functions. A rule of thumb is that routines

which write to the screen, or check the status of or read the keyboard, and any disk I/O routines use DOS and should be avoided in your ISR.

The same problem of reentrancy exists for many floating point emulators as well, meaning you may have to avoid floating point (real) math in your ISR.

Note that the problem of reentrancy exists, no matter what programming language you are using. Even if you are writing your ISR in assembly language, DOS and many floating point emulators are not reentrant. Of course, there are ways around this problem, such as those which involve checking to see if any DOS functions are currently active when your ISR is called, but such solutions are well beyond the scope of this discussion.

The second major concern when writing your ISR is to make it as short as possible in terms of execution time. Spending long periods of time in your ISR may mean that other important interrupts are being ignored. Also, if you spend too long in your ISR, it may be called again before you have completed handling the first run. This often leads to a hang that requires a reboot.

Your ISR should have this structure:

- Push any processor registers used in your ISR. Most C and Pascal interrupt routines automatically do this for you.
- Put the body of your routine here.
- Issue the EOI command to the 8259 interrupt controller by writing 20H to port 20H and port A0H (if you are using IRQ8-IRQ15).
- Pop all registers pushed on entrance. Most C and Pascal interrupt routines automatically do this for you.

The following C and Pascal examples show what the shell of your ISR should be like:

**In C:**

```
void interrupt ISR(void)
{
    /* Your code goes here. Do not use any DOS functions! */
    outportb(0x20, 0x20);              /* Send EOI command to 8259A (for all IRQs)*/
    outportb(0x20, 0xA0);              /* Send EOI command to 8259B (if using IRQ8-15) */
}
```

**In Pascal:**

```
Procedure ISR; Interrupt;
begin
    { Your code goes here. Do not use any DOS functions! }
    Port[$20] := $20;                  { Send EOI command to 8259A (for all IRQs) }
    Port[$A0] := $20;                  { Send EOI command to 8259B (if using IRQ8-15) }
end;
```

**• Saving the Startup Interrupt Mask Register (IMR) and Interrupt Vector**

The next step after writing the ISR is to save the startup state of the interrupt mask register and the interrupt vector that you will be using. The IMR for IRQ0-IRQ7 is located at I/O port 21H; the IMR for IRQ8-IRQ15 is located at I/O port A1H. The interrupt vector you will be using is located in the interrupt vector table which is simply an array of 256 four-byte pointers and is located in the first 1024 bytes of memory (Segment = 0, Offset = 0). You can read this value directly, but it is a better practice to use DOS function 35H (get interrupt vector). Most C and Pascal compilers provide a library routine for reading the value of a vector. The vectors for IRQ0-IRQ7 are vectors 8 through 15, where IRQ0 uses vector 8, IRQ1 uses vector 9, and so on. The vectors for IRQ8-IRQ15 are vectors 70H through 77H, where IRQ8 uses vector 70H, IRQ9 uses vector 71H, and so on. Thus, if the TC1024 will be using IRQ15, you should save the value of interrupt vector 77H.

Before you install your ISR, temporarily mask out the IRQ you will be using. This prevents the IRQ from requesting an interrupt while you are installing and initializing your ISR. To mask the IRQ, read in the current IMR at I/O port 21H for IRQ0-IRQ7, or at I/O port A1H for IRQ8-IRQ15 and **set** the bit that corresponds to your IRQ

(remember, setting a bit disables interrupts on that IRQ while clearing a bit enables them). The IMR on 8259A is arranged so that bit 0 is for IRQ0, bit 1 is for IRQ1, and so on. The IMR on 8259B is arranged so that bit 0 is for IRQ8, bit 1 is for IRQ9, and so on. See the paragraph entitled *Interrupt Mask Register (IMR)* earlier in this chapter for help in determining your IRQ's bit. After setting the bit, write the new value to I/O port 21H (IRQ0-IRQ7) or I/O port A1H (IRQ8-IRQ15).

With the startup IMR saved and the interrupts on your IRQ temporarily disabled, you can assign the interrupt vector to point to your ISR. Again, you can overwrite the appropriate entry in the vector table with a direct memory write, but this is a bad practice. Instead, use either DOS function 25H (set interrupt vector) or, if your compiler provides it, the library routine for setting an interrupt vector. Remember that vectors 8-15 are for IRQ0-IRQ7 and vectors 70H-77H are for IRQ8-IRQ15.

If you need to program the source of your interrupts, do that next. For example, if you are using a timer/counter to generate interrupts, you must program it to run in the proper mode and at the proper rate.

Finally, clear the bit in the IMR for the IRQ you are using. This enables interrupts on the IRQ.

**• Restoring the Startup IMR and Interrupt Vector**

Before exiting your program, you must restore the interrupt mask register and interrupt vectors to the state they were in before your program started. To restore the IMR, write the value that was saved when your program started to I/O port 21H for IRQ0-IRQ7 or I/O port A1H for IRQ8-IRQ15. Restore the interrupt vector that was saved at startup with either DOS function 25H (set interrupt vector), or use the library routine supplied with your compiler. Performing these two steps will guarantee that the interrupt status of your computer is the same after running your program as it was before your program started running.

**• Common Interrupt Mistakes**

- Remember that hardware interrupts are numbered 8 through 15 for IRQ0-IRQ7 and 70H through 77H for IRQ8-IRQ15.
- One of the most common mistakes when writing an ISR is forgetting to issue the EOI command to the appropriate 8259 interrupt controller before exiting the ISR.

## Example Programs

Included with the TC1024 is a set of example programs that demonstrate the use of many of the board's features. These examples are in written in C and Pascal. Also included is an easy-to-use menu-driven diagnostics program, 1024DIAG, which is especially helpful when you are first checking out your board after installation.

### C and Pascal Programs

These programs are source code files so that you can easily develop your own custom software for your TC1024 board.

**Timer/Counter:**

| | |
|---|---|
| INTRPTS | Shows how to generate interrupts and read the digital I/O lines. |
| COUNT | Shows how to use the Am9513A as a simple counter. |

**Digital I/O:**

| | |
|---|---|
| DIGITAL | Simple program that shows how to read and write the digital I/O lines. |

# CHAPTER 5

## EXAMPLES OF Am9513A APPLICATIONS

This chapter steps through an example program to help you understand how the Am9513A registers are programmed. The data pointer register and command registers are summarized in tables. The master mode and counter mode register bit assignments are also included, as well as the frequency scaler ratios.

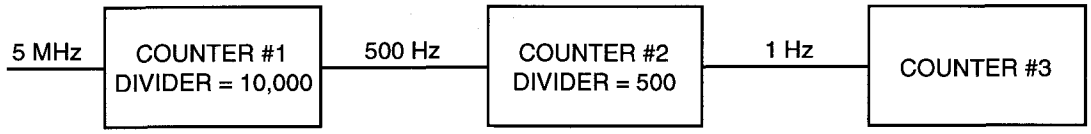
This chapter provides a more detailed look at an example program using the Am9513A for counting. If you are unfamiliar with the Am9513A and how it is programmed, walking through this example and the other example programs included on your TC1024 disk may be the best way to understand the many registers and their operation so that you can successfully develop your own programs for your specific applications.

**IMPORTANT**

Because of the bus release time of the Am9513A, AMD recommends you insert a small delay between software accesses to the chip.

## EXAMPLE: Counting Program Using Timer/Counters 1, 2, and 3

This Turbo C program, EXAMPLE.C on the included disk, shows you how to program the Am9513A's timer/counters 1, 2, and 3 to perform a simple counting function. In this example, counter 1 is used to divide the on-board 5 MHz clock by 10,000. The output from counter 1 (5 MHz ÷ 10,000 = 500 Hz) is used to clock counter 2. Counter 2 is used to divide this 500 Hz clock by 500. The result is a 1 Hz clock which is used to clock counter 3. Counter 3 counts the 1 Hz pulses. The count value from counter 3 is displayed on the screen. This value should start at 0 and increment once each second.




The first lines of the program initialize the board. The address in the variable "BA" must match the setting of the base address switch, S1, on the board. The factory setting of S1 is 300 hex (768 decimal).

```
int board, j, result, dr, cr;
board = 768;
dr = board + 12;
cr = board + 14;
```

Now, reset the Am9513A timer/counter chip (see Table 5-2):

```
outport (cr,0xff);                'AM9513A MASTER RESET
```

Next, set up the Am9513A master mode register (see Figure 5-1). These are the settings we will use:

Scaler Control = binary division
Data Pointer Control = disable increment
Data Bus Width = 16 bits
FOUT Gate = FOUT on
FOUT Divider = divide by 16
FOUT Source = F1 (see Figure 5-3)
Compare 2 Enable = disabled
Compare 1 Enable = disabled
Time-of-Day Mode = disabled

VALUE = HEX 6000

```
outport (cr,0xff17);              /* point to master mode register (table 5-1) */
outport (cr,0x00);                /* master mode lsb */
outport (cr,0x60);                /* master mode msb */
```

| Table 5-1 — Load Data Pointer Commands | | | | |
|---|---|---|---|---|
| | Element Cycle | | | Hold Cycle |
| | Mode Register | Load Register | Hold Register | Hold Register |
| Counter 1 | FF01 | FF09 | FF11 | FF19 |
| Counter 2 | FF02 | FF0A | FF12 | FF1A |
| Counter 3 | FF03 | FF0B | FF13 | FF1B |
| Counter 4 | FF04 | FF0C | FF14 | FF1C |
| Counter 5 | FF05 | FF0D | FF15 | FF1D |
| Master Mode Register = FF17<br>Alarm 1 Register = FF07<br>Alarm 2 Register = FF0F<br>Status Register = FF1F | | | | |

Next, set up the counter 1 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = no gating
Source Edge = rising edge
Count Source Selection = F1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count down
Output Control = TC toggled

VALUE = HEX 0B22

```
outport (cr,0xff01);            /* point to counter 1 mode register (table 5-1) */
outport (dr,0x0b22);            /* counter 1 mode */
```

Put the hex number 2710 (decimal 10,000) in counter 1 load register:

```
outport (cr,0xff09);            /* point to counter 1 load register (table 5-1) */
outport (dr,0x2710);            /* counter 1 data */
```

Next, set up the counter 2 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = no gating
Source Edge = rising edge
Count Source Selection = TCN-1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count down
Output Control = TC toggled

VALUE = HEX 0022

| Table 5-2 — Am9513A Command Summary | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Command Code | | | | | | | | Command Description |
| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 | |
| 0 | 0 | 0 | E2 | E1 | G4 | G2 | G1 | Load data pointer register with contents of E & G fields. (E ≠ 000, G ≠ 110). E & G fields described in Appendix C. |
| 0 | 0 | 1 | S5 | S4 | S3 | S2 | S1 | Arm counting for all selected counters |
| 0 | 1 | 0 | S5 | S4 | S3 | S2 | S1 | Load contents of specified source into all selected counters |
| 0 | 1 | 1 | S5 | S4 | S3 | S2 | S1 | Load & arm all selected counters* |
| 1 | 0 | 0 | S5 | S4 | S3 | S2 | S1 | Disarm & save all selected counters |
| 1 | 0 | 1 | S5 | S4 | S3 | S2 | S1 | Save all selected counters in hold register |
| 1 | 1 | 0 | S5 | S4 | S3 | S2 | S1 | Disarm all selected counters |
| 1 | 1 | 1 | 0 | 1 | N4 | N2 | N1 | Set toggle out (high) for counter N (001 ≤ N ≤ 101) |
| 1 | 1 | 1 | 0 | 0 | N4 | N2 | N1 | Clear toggle out (low) for counter N (001 ≤ N ≤ 101) |
| 1 | 1 | 1 | 1 | 0 | N4 | N2 | N1 | Step counter N (001 ≤ N ≤ 101) |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | Set MM14 (disable data pointer sequencing) |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | Set MM12 (gate off FOUT) |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | Set MM13 (enter 16-bit bus mode) |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Clear MM14 (enable data pointer sequencing) |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | Clear MM12 (gate on FOUT) |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | Clear MM13 (enter 8-bit bus mode) |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | Enable prefetch for write operations |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | Disable prefetch for write operations |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | Master reset |
| * Not to be used for asynchronous operations. | | | | | | | | |

```
outport (cr,0xff02);            /* point to counter 2 mode register (table 5-1) */
outport (dr,0x0022);            /* counter 2 mode */
```

Put the hex number 1F4 (decimal 500) in counter 2 load register:

```
outport (cr,0xff0a);            /* point to counter 2 load register (table 5-1) */
outport (dr,0x01f4);            /* counter 2 data */
```

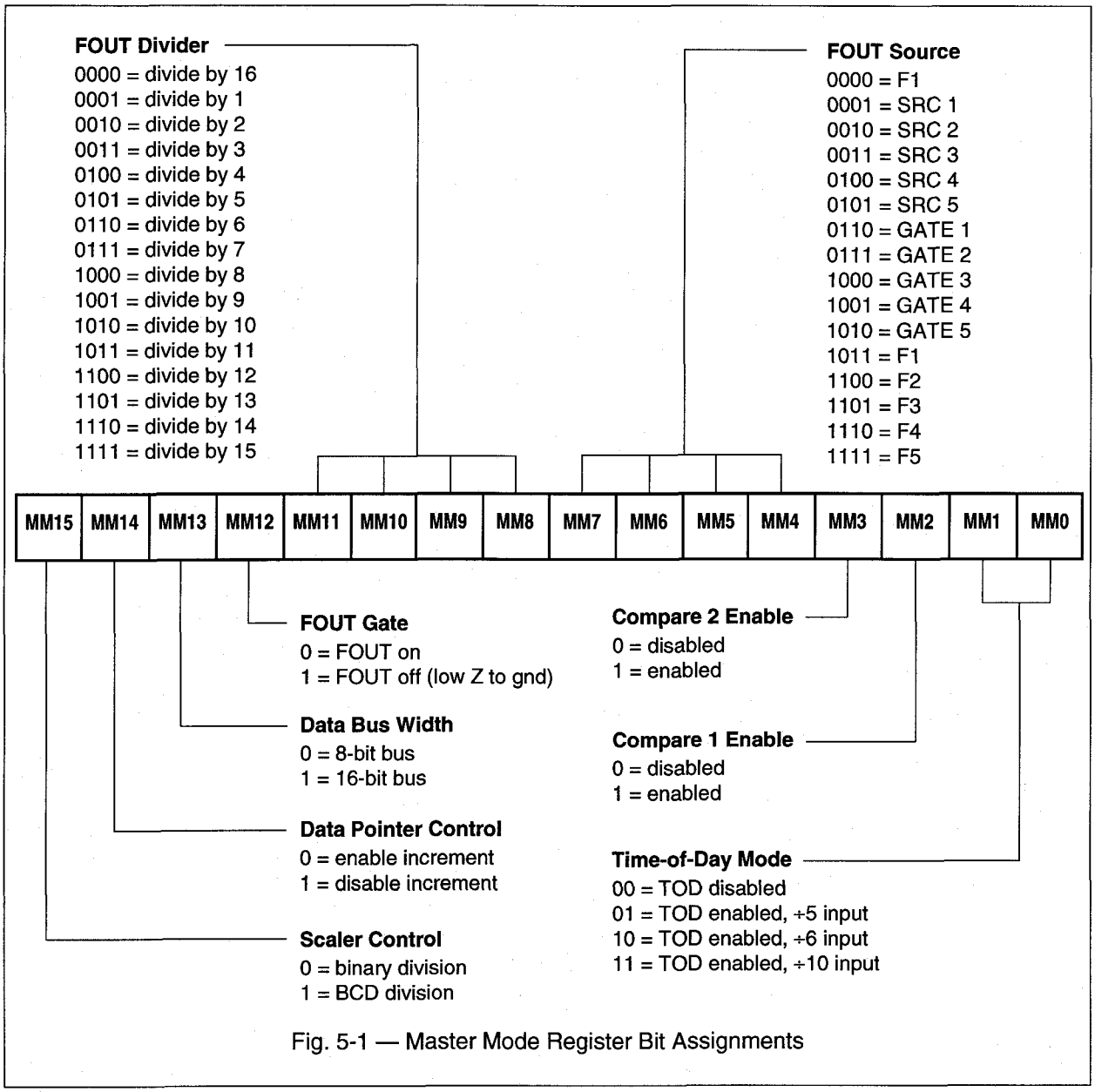



Fig. 5-1 — Master Mode Register Bit Assignments

Next, set up the counter 3 mode register (see Figure 5-2). These are the settings we will use:

Gating Control = no gating
Source Edge = rising edge
Count Source Selection = TCN-1
Count Control = disable special gate
= reload from load
= count repetitively
= binary count
= count up
Output Control = TC toggled

VALUE = HEX 002A

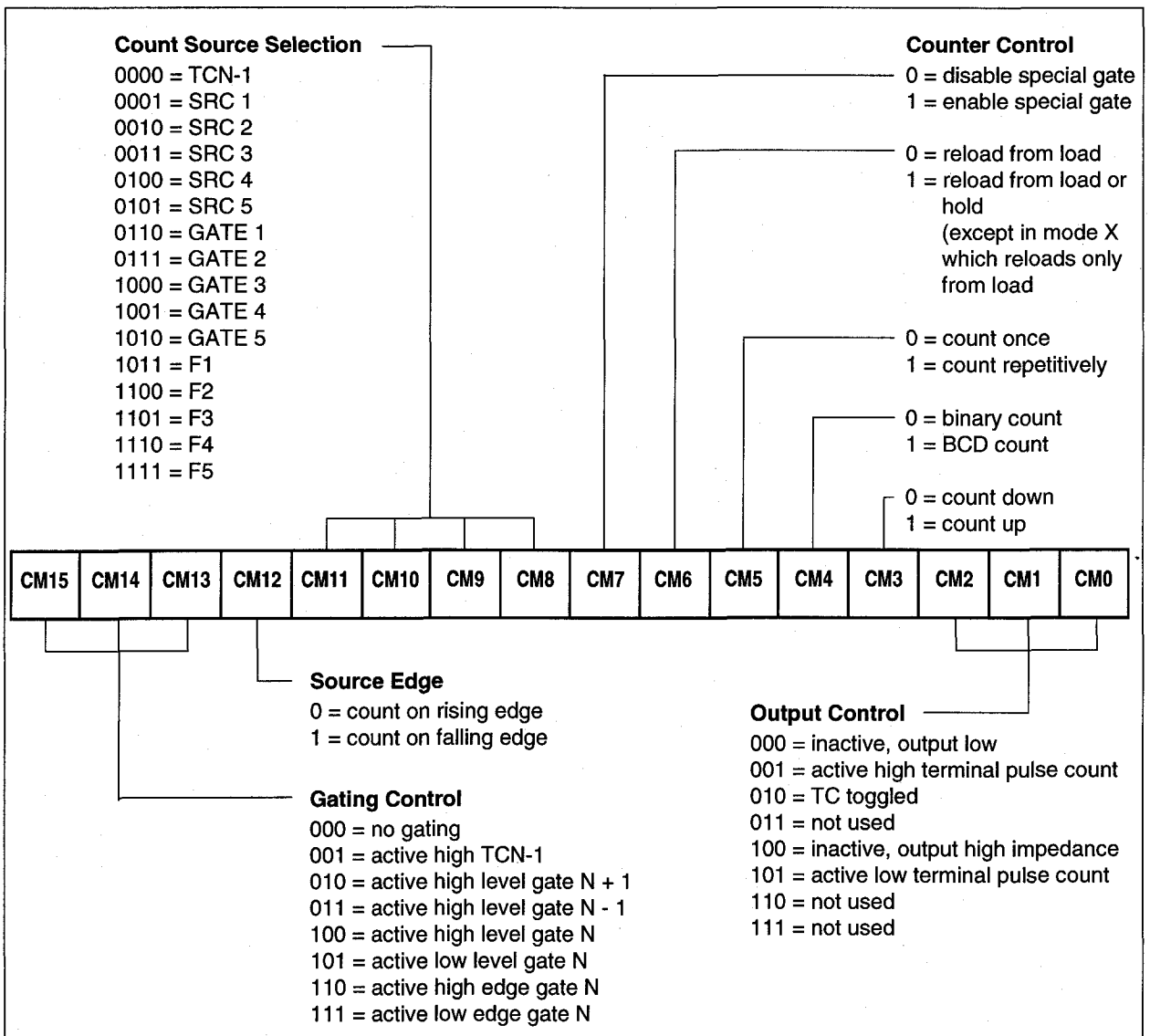
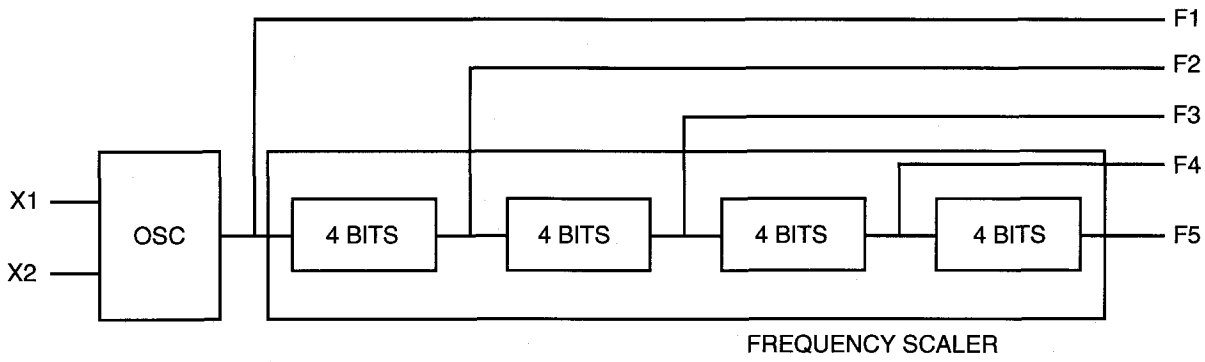



Fig. 5-2 — Counter Mode Register Bit Assignments

```
outport (cr,0xff03);          /* point to counter 3 mode register (table 5-1) */
outport (dr,0x002a);          /* counter 3 mode */
```

Put the hex number 0000 in counter 3 load register:

```
outport (cr,0xff0b);          /* point to counter 3 load register (table 5-1) */
outport (dr,0x0000);          /* counter 3 data */
outport (cr,0x0067);          /* load and arm counters 1, 2 & 3 (table 5-2) */
```




| Frequency | BCD Scaling (MM15 = 1) | | Binary Scaling (MM15 = 0) | |
|---|---|---|---|---|
| | Ratio | With On-board 5 MHz Clock | Ratio | With On-board 5 MHz Clock |
| F1 | OSC | 5 MHz | OSC | 5 MHz |
| F2 | F1 ÷ 10 | 500 kHz | F1 ÷ 16 | 312.5 kHz |
| F3 | F1 ÷ 100 | 50 kHz | F1 ÷ 256 | 19.53 kHz |
| F4 | F1 ÷ 1000 | 5 kHz | F1 ÷ 4096 | 1.221 kHz |
| F5 | F1 ÷ 10,000 | 500 Hz | F1 ÷ 65,536 | 76.3 Hz |

Fig. 5-3 — Frequency Scaler Ratio

The main program for taking a total of 25 readings is:

```
j =0;
clrscr();
while (j<25)
  {
    outport(cr,0x00a4);          /* save counter 3 in hold register (table 5-2) */
    outport(cr,0xff13);          /* point to counter 3 hold register (table 5-1) */
    result=inport(dr);           /* read counter 3 data */
    printf("%5d",result);        /* print result */
    delay(1000);
    j = j+1;

  }
```

# APPENDIX A

## TC1024 SPECIFICATIONS

## TC1024 Characteristics Typical @ 25° C

### Interface

AT bus compatible
Switch-selectable base address, I/O mapped
Jumper-selectable interrupts

**Digital I/O ..... CMOS 82C55 (Optional NMOS 8255)**

Number of lines ..... 24
Logic compatibility ..... TTL/CMOS (Configurable with optional I/O pull-up/pull-down resistors)
High-level output voltage ..... 4.2V, min
Low-level output voltage ..... 0.45V, max
High-level input voltage ..... 2.2V, min; 5.5V, max
Low-level input voltage ..... -0.3V, min; 0.8V, max
High-level output current, Isource ..... CMOS buffer: -12 mA, max; TTL buffer: -16 mA, max
Low-level output current, Isink ..... CMOS buffer: 24 mA, max; TTL buffer: 64 mA, max
Input load current ..... ±10 µA
Input capacitance, C(IN)@F=1MHz ..... 10 pF
Output capacitance, C(OUT)<@F=1MHz ..... 20 pF

**Timer/Counter ..... Am9513A**

Ten 16-bit timer/counters (2 Am9513A chips)
Binary or BCD up or down counting
Programmable operating modes ..... 24
Counter input source ..... External clock (6.9 MHz, max); on-board 5 MHz clock; external gate input; or adjacent counter output
Counter outputs ..... Available externally; used as PC interrupts or internally cascaded to adjacent counter
Counter gate source ..... External input; counter output; or software control

### Miscellaneous Inputs/Outputs

+5 volts, ±12 volts, digital ground (PC bus-sourced)
External interrupt input
Frequency output

### Current Requirements

350 mA @ +5 volts

### Connectors

P3: 50-pin right angle shrouded box header
P4: 20-pin box connector

### Environmental

Operating temperature ..... 0 to +70°C
Storage temperature ..... -40 to +85°C
Humidity ..... 0 to 90% non-condensing

### Size

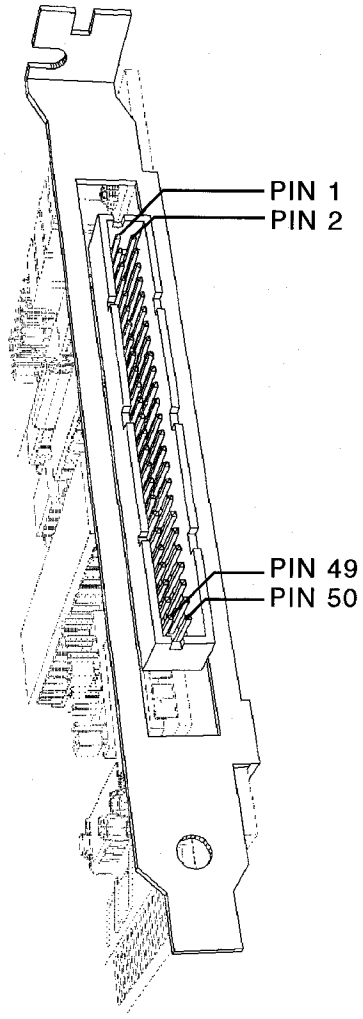
3.875"H x 6.370"L (99mm x 162mm)

# APPENDIX B

## P3 AND P4 CONNECTOR PIN ASSIGNMENTS

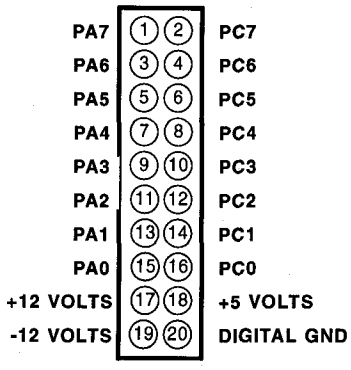
**P3 Connector:**

| | | | |
|---|---|---|---|
| SRC1 | 1 | 2 | SRC6 |
| GATE1 | 3 | 4 | GATE6 |
| OUT1 | 5 | 6 | OUT6 |
| SRC2 | 7 | 8 | SRC7 |
| GATE2 | 9 | 10 | GATE7 |
| OUT2 | 11 | 12 | OUT7 |
| SRC3 | 13 | 14 | SRC8 |
| GATE3 | 15 | 16 | GATE8 |
| OUT3 | 17 | 18 | OUT8 |
| SRC4 | 19 | 20 | SRC9 |
| GATE4 | 21 | 22 | GATE9 |
| OUT4 | 23 | 24 | OUT9 |
| SRC5 | 25 | 26 | SRC10 |
| GATE5 | 27 | 28 | GATE10 |
| OUT5 | 29 | 30 | OUT10 |
| EXTINT | 31 | 32 | DIGITAL GND |
| FOUT | 33 | 34 | DIGITAL GND |
| PC3 | 35 | 36 | PC2 |
| PC1 | 37 | 38 | PC0 |
| PB7 | 39 | 40 | PB6 |
| PB5 | 41 | 42 | PB4 |
| PB3 | 43 | 44 | PB2 |
| PB1 | 45 | 46 | PB0 |
| +12 VOLTS | 47 | 48 | +5 VOLTS |
| -12 VOLTS | 49 | 50 | DIGITAL GND |




| P3 Mating Connector Part Numbers | |
|---|---|
| **Manufacturer** | **Part Number** |
| AMP | 1-746094-0 |
| 3M | 3425-7650 |

**P4 Connector:**




| P4 Mating Connector Part Numbers | |
|---|---|
| **Manufacturer** | **Part Number** |
| AMP | 1-746094-4 |

# APPENDIX C

## COMPONENT DATA SHEETS

# AMD Am9513A System Timing Controller
# Data Sheet Reprint

# Chapter 1
# The Am9513A/Am9513

## INTRODUCTION

Manipulation and coordination of timing parameters and event sequences are universal system attributes. At the most fundamental levels of control, time sequences are intimately embedded in the essential hardware and interface concepts of all processors: the necessary flows of step-by-step procedures are inherent in the execution of even the most basic programs. At the interface level, both internal and external hardware coordination usually require several types of timing-oriented exchanges. In general, control of system and sub-system processes will often involve sophisticated levels of counting, sequencing and timing manipulations. The specific mix of such activities will, of course, be application dependent, yet counting/timing concepts are at least fundamentally involved in all system operations, from the simplest sequencing of a hardware interface to the complex interaction of high-level processes.

Time-related activities fall into a wide variety of categories. Frequency generation, waveform duty cycle control, event counting, interval measurement, precise periodic interrupts, time-of-day accumulation, delays, gap detection, etc., are just a few of the types of operations typically undertaken. When the system must accomplish several of these activities, especially when some measure of concurrency is necessary, a significant portion of the available processing and/or hardware logic resources can be consumed. Throughput limitations easily arise.

A specialized circuit with enough versatility to handle many types of counting and timing functions would therefore be able to simplify software, improve system performance and decrease system chip count. The Am9513 System Timing Controller has been designed to accomplish just such a task. It provides significant capability for waveform generation, counting, timing and intervalometer functions for many types of processor-oriented systems. It offers an unusually versatile control structure that allows the use of many operating configurations so that a wide variety of applications can be efficiently serviced.

The operating philosophy of the Am9513 is based on the use of general-purpose counters that can be controlled in various ways to produce the functions desired. Broadly, use of the counters falls into two classic categories: (a) count accumulation, and (b) frequency division.

In the first case, the counter simply accumulates a count of transitions that occur on its input. An output that indicates the zero state of the counter would be of only incidental interest. The counter value should be available at any time to the associated CPU or it might be compared with some independent value. The accumulated count might be modified or the counter input conditioned by various controls, including hardware and software gating functions; in any event, in these types of applications, it is the value of the actual count that is of interest.

In the case of frequency division, on the other hand, it is an output waveform that is of interest and the counter input information may be incidental. With an output signal that indicates the zero state of the counter, selection of the effective length of the counter and the input frequency are controlled to provide the desired output frequency. Additional controls may allow various types of output waveforms to be generated from the base output frequency, but the actual counter value will usually not be of direct interest.

The Am9513 has been designed to handle effectively both modes of operation, even intermixed on the same chip. In many instances, of course, both types of counter usage will be combined to provide the desired function.

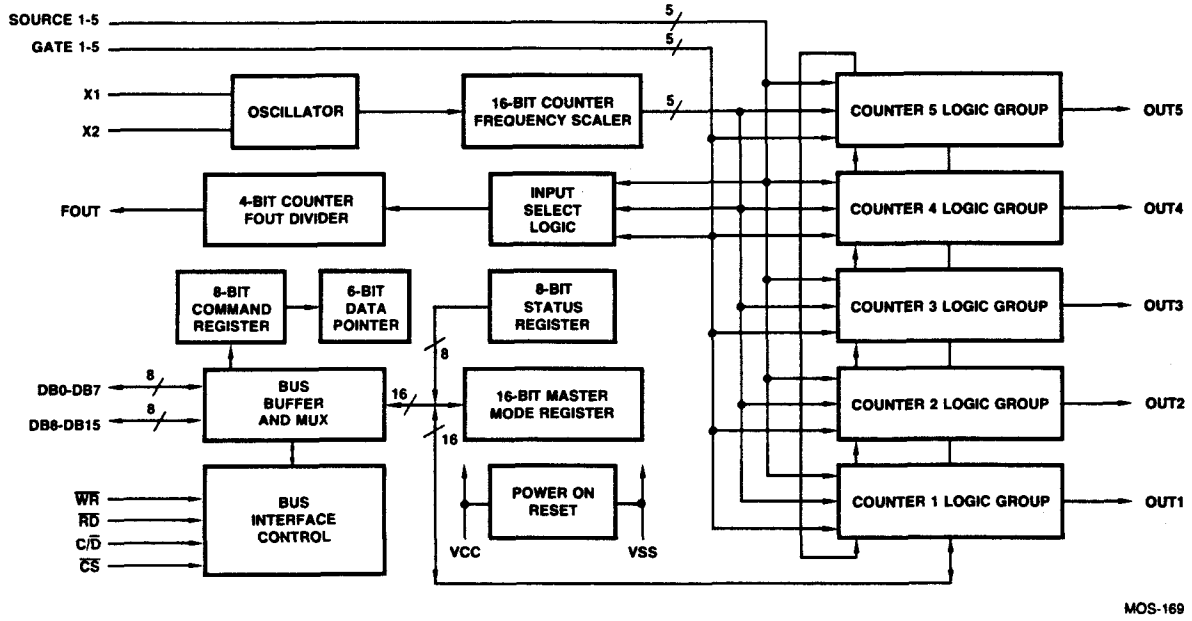



**Figure 1-1. General Block Diagram**

## FUNCTIONAL DESCRIPTION

The Am9513 System Timing Controller (STC) is a support device for processor oriented systems that is designed to enhance the available capability with respect to counting and timing operations. It provides the capability for programmable frequency synthesis, high resolution programmable duty cycle waveforms, retriggerable digital timing functions, time-of-day clocking, coincidence alarms, complex pulse generation, high resolution baud rate generation, frequency shift keying, stop-watching timing, event count accumulation, waveform analysis and many more. A variety of programmable operating modes and control features allow the Am9513 to be personalized for particular applications as well as dynamically reconfigured under program control.

The STC includes five general-purpose 16-bit counters. A variety of internal frequency sources and external pins may be selected as inputs for individual counters with software selectable active-high or active-low input polarity. Both hardware and software gating of each counter is available. Three-state outputs for each counter provide either pulses or levels. The counters can be programmed to count up or down in either binary or BCD. The accumulated count may be read without disturbing the counting process. Any of the counters may be internally concatenated to form an effective counter length of up to 80 bits.

The Am9513 block diagrams (Figures 1-1, 1-2 and 1-3) indicate the interface signals and the basic flow of information. Internal control lines and the internal data bus have been omitted. The control and data registers are all connected to a common internal 16-bit bus. The external bus may be 8- or 16-bits wide; in the 8-bit mode the internal 16-bit information is multiplexed to the low order data bus pins DB0 through DB7.

An internal oscillator provides a convenient source of frequencies for use as counter inputs. The oscillator's frequency is controlled at the X1 and X2 interface pins by an external reactive network such as a crystal. The oscillator output is divided by the Frequency Scaler to provide several sub-frequencies. One of the scaled frequencies (or one of ten input signals) may be selected as an input to the FOUT divider and then comes out of the chip at the FOUT interface pin.

The STC is addressed by the external system as two locations: a Control port and a Data port. The Control port provides direct access to the Status and Command registers, as well as allowing the user to update the Data Pointer register. The Data port is used to communicate with all other addressable internal locations. The Data Pointer register controls the Data port addressing.

Among the registers accessible through the Data port are the Master Mode register and five Counter Mode registers, one for each counter. The Master Mode register controls the programmable options that are not controlled by the Counter Mode registers.

Each of the five general-purpose counters is 16-bits long and is independently controlled by its Counter Mode register. Through this register, a user can software select one of 16 sources as the counter input, a variety of gating and repetition modes, up or down counting in binary or BCD and active-high or active-low input and output polarities.

Associated with each counter are a Load register and a Hold register, both accessible through the Data port. The Load register is used to automatically reload the counter to any predefined value, thus controlling the effective count period. The Hold register is used to save count values without disturbing the count process, permitting the host processor to read intermediate counts. In addition, the Hold register may be used as a second Load register to generate a number of complex output waveforms.

All five counters have the same basic control logic and control registers. Counters 1 and 2 have additional Alarm registers and comparators associated with them, plus the extra logic necessary for operating in a 24-hour time-of-day mode. For real-time operation the time-of-day logic will accept 50Hz, 60Hz or 100Hz input frequencies.

Each general counter has a single dedicated output pin. It may be turned off when the output is not of interest or may be configured in a variety of ways to drive interrupt controllers, Darlington buffers, bus drivers, etc. The counter inputs, on the other hand, are specifically not dedicated to any given interface line. Considerable versatility is available for configuring both the input and the gating of individual counters. This not only permits dynamic reassignment of inputs under software control, but also allows multiple counters to use a single input, and allows a single gate pin to control more than one counter. Indeed, a single pin can be the gate for one counter and, at the same time, the count source for another.

A powerful command structure simplifies user interaction with the counters. A counter must be armed by one of the ARM commands before counting can commence. Once armed, the counting process may be further enabled or disabled using the hardware gating facilities. The ARM and DISARM commands permit software gating of the count process in some modes.

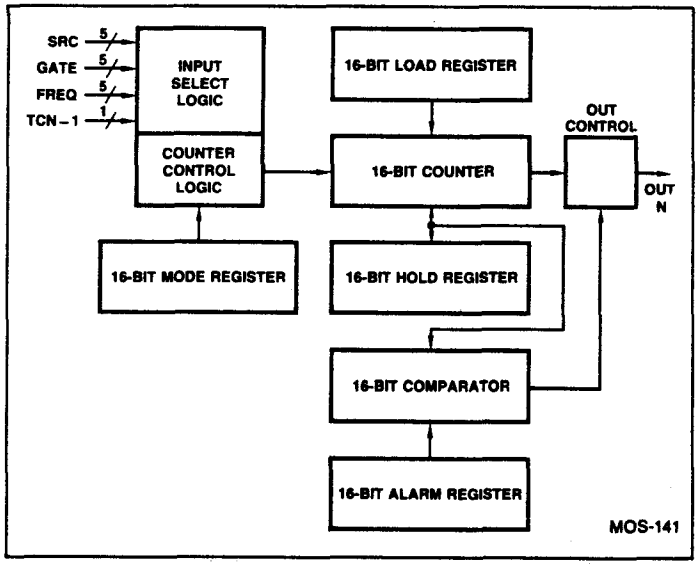



**Figure 1-2. Counter Logic Groups 1 and 2**

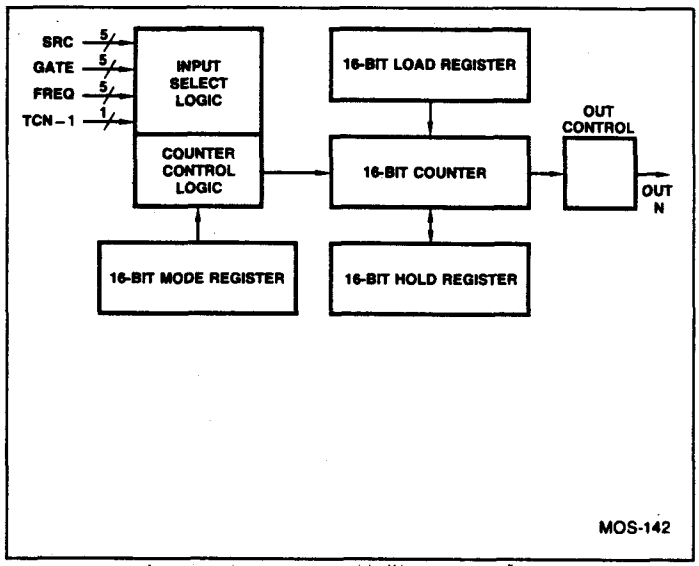



**Figure 1-3. Counter Logic Groups 3, 4 and 5**

The LOAD command causes the counter to be reloaded with the value in either the associated Load register or the associated Hold register. It will often be used as a software retrigger or as counter initialization prior to active hardware gating.

The DISARM command disables further counting independent of any hardware gating. A disarmed counter may be reloaded using the LOAD command, may be incremented or decremented using the STEP command and may be read using the SAVE command. A count process may be resumed using an ARM command.

The SAVE command transfers the contents of a counter to its associated Hold register. This command will overwrite any previous Hold register contents. The SAVE command is designed to allow an accumulated count to be preserved so that it can be read by the host CPU at some later time.

Two combinations of the basic commands exist to either LOAD AND ARM or to DISARM AND SAVE any combination of counters. Additional commands are provided to: step an individual counter by one count; set and clear an output toggle; issue a software reset; clear and set special bits in the Master Mode register; and load the Data Pointer register.

Note: Separate LOAD and ARM commands should be used for asynchronous operations.

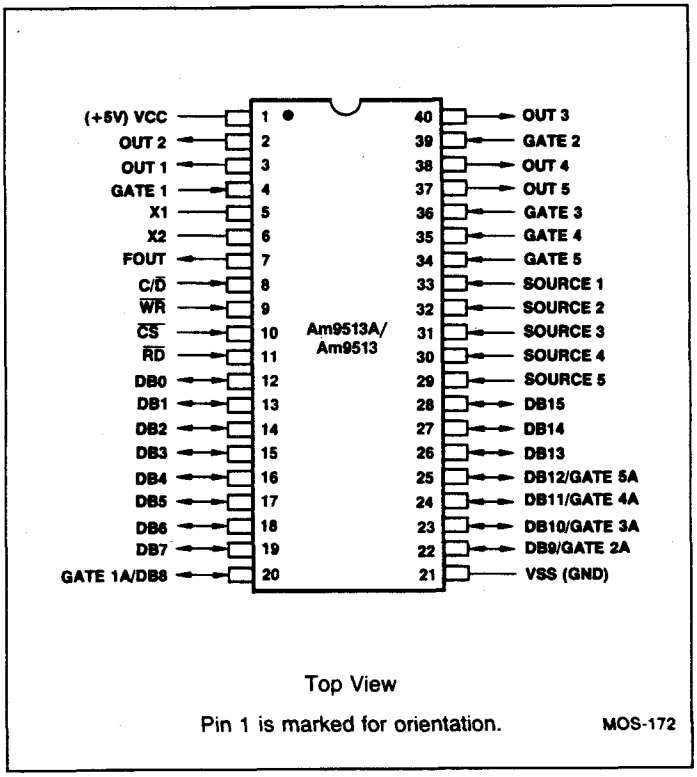



**Figure 1-4. Connection Diagram**

## INTERFACE SIGNAL DESCRIPTION

Figure 1-5 summarizes the interface signals and their abbreviations for the STC. Figure 1-4 shows the signal pin assignments for the standard 40-pin dual in-line package.

**VCC:** +5 volt power supply

**VSS:** Ground

### X1, X2 (Crystal)

X1 and X2 are the connections for an external crystal used to determine the frequency of the internal oscillator. The crystal should be a parallel-resonant, fundamental-mode type. An RC or LC or other reactive network may be used instead of a crystal. For driving from an external frequency source, X1 should be left open and X2 should be connected to a TTL source and a pull-up resistor.

### FOUT (Frequency Out, Output)

The FOUT output is derived from a 4-bit counter that may be programmed to divide its input by any integer value from 1 through 16 inclusive. The input to the counter is selected from any of 15 sources, including the internal scaled oscillator frequencies. FOUT may be gated on and off under software control and when off will exhibit a low impedance to ground. Control over the various FOUT options resides in the Master Mode register. After power-up, FOUT provides a frequency that is 1/16 that of the oscillator.

### GATE1-GATE5 (Gate, Inputs)

The Gate inputs may be used to control the operations of individual counters by determining when counting may proceed. The same Gate input may control up to three counters. Gate pins may also be selected as count sources for any of the counters and for the FOUT divider. The active polarity for a selected Gate input is programmed at each counter. Gating function options allow level-sensitive gating or edge-initiated gating. Other gating modes are available including one that allows the Gate input to select between two counter output frequencies. All gating functions may also be disabled. The active Gate input is conditioned by an auxiliary input when the unit is operating with an external 8-bit data bus. See Data Bus description. Schmitt-trigger circuitry on the GATE inputs allows slow transition times to be used.

| Signal | Abbreviation | Type | Pins |
|---|---|---|---|
| +5 Volts | VCC | Power | 1 |
| Ground | VSS | Power | 1 |
| Crystal | X1, X2 | I/O, I | 2 |
| Read | $\overline{RD}$ | Input | 1 |
| Write | $\overline{WR}$ | Input | 1 |
| Chip Select | $\overline{CS}$ | Input | 1 |
| Control/Data | C/$\overline{D}$ | Input | 1 |
| Source N | SRC | Input | 5 |
| Gate N | GATE | Input | 5 |
| Data Bus | DB | I/O | 16 |
| Frequency Out | FOUT | Output | 1 |
| Out N | OUT | Output | 5 |

**Figure 1-5. Interface Signal Summary**

### SRC1-SRC5 (Source, Inputs)

The Source inputs provide external signals that may be counted by any of the counters. Any Source line may be routed to any or all of the counters and the FOUT divider. The active polarity for a selected SRC input is programmed at each counter. Any duty cycle waveform will be accepted as long as the minimum pulse width is at least half the period of the maximum specified counting frequency for the part. Schmitt-trigger circuitry on the SRC inputs allows slow transition times to be used.

**OUT1-OUT5 (Counter, Outputs)**

Each 3-state OUT signal is directly associated with a corresponding individual counter. Depending on the counter configuration, the OUT signal may be a pulse, a square wave, or a complex duty cycle waveform. OUT pulse polarities are individually programmable. The output circuitry detects the counter state that would have been all bits zero in the absence of a reinitialization. That information is used to generate the selected waveform type. An optional output mode for Counters 1 and 2 overrides the normal output mode and provides a true OUT signal when the counter contents match the contents of an Alarm register.

**DB0-DB7, DB8-DB15 (Data Bus, Input/Output)**

The 16, bidirectional Data Bus lines are used for information exchanges with the host processor. HIGH on a Data Bus line corresponds to one and LOW corresponds to zero. These lines act as inputs when $\overline{WR}$ and $\overline{CS}$ are active and as outputs when $\overline{RD}$ and $\overline{CS}$ are active. When $\overline{CS}$ is inactive, these pins are placed in a high-impedance state.

After power-up or reset, the data bus will be configured for 8-bit width and will use only DB0 through DB7. DB0 is the least significant and DB7 is the most significant bit position. The data bus may be reconfigured for 16-bit width by changing a control bit in the Master Mode register. This is accomplished by writing an 8-bit command into the low-order DB lines while holding the DB13-DB15 lines at a logic high level. Thereafter all 16 lines can be used, with DB0 as the least significant and DB15 as the most significant bit position.

When operating in the 8-bit data bus environment, DB8-DB15 will never be driven active by the Am9513. DB8 through DB12 may optionally be used as additional Gate inputs (see Figure 1-6). If unused they should be held high. When pulled low, a GATENA signal will disable the action of the corresponding counter N gating. DB13-DB15 should be held high in 8-bit bus mode whenever $\overline{CS}$ and $\overline{WR}$ are simultaneously active.

**$\overline{CS}$ (Chip Select, Input)**

The active-low Chip Select input enables Read and Write operations on the data bus. When Chip Select is high, the Read and Write inputs are ignored. The first Chip Select signal after power-up is used to clear the power-on reset circuitry. If Chip Select is tied to ground permanently, the power-on reset circuitry may not function. In such a configuration, the software reset command must be issued following power-up to reset the Am9513.

**$\overline{RD}$ (Read, Input)**

The active-low Read signal is conditioned by Chip Select and indicates that internal information is to be transferred to the data bus. The source will be determined by the port being addressed and, for Data Port reads, by the contents of the Data Pointer register. $\overline{WR}$ and $\overline{RD}$ should be mutually exclusive.

**$\overline{WR}$ (Write, Input)**

The active-low Write signal is conditioned by Chip Select and indicates that data bus information is to be transferred to an internal location. The destination will be determined by the port being addressed and, for Data Port writes, by the contents of the Data Pointer register. $\overline{WR}$ and $\overline{RD}$ should be mutually exclusive.

**C/$\overline{D}$ (Control/Data, Input)**

The Control/Data signal selects source and destination locations for read and write operations on the data bus. Control Write operations load the Command register and the Data Pointer. Control Read operations output the Status register. Data Read and Data Write transfers communicate with all other internal registers. Indirect addressing at the data port is controlled internally by the Data Pointer register.

| Package Pin | Data Bus Width (MM14) | |
|---|---|---|
| | 16 Bits | 8 Bits |
| 12 | DB0 | DB0 |
| 13 | DB1 | DB1 |
| 14 | DB2 | DB2 |
| 15 | DB3 | DB3 |
| 16 | DB4 | DB4 |
| 17 | DB5 | DB5 |
| 18 | DB6 | DB6 |
| 19 | DB7 | DB7 |
| 20 | DB8 | GATE 1A |
| 22 | DB9 | GATE 2A |
| 23 | DB10 | GATE 3A |
| 24 | DB11 | GATE 4A |
| 25 | DB12 | GATE 5A |
| 26 | DB13 | (VIH) |
| 27 | DB14 | (VIH) |
| 28 | DB15 | (VIH) |

**Figure 1-6. Data Bus Assignments**

**Interface Considerations**

All of the input and output signals for the Am9513 are specified with logic levels compatible with those of standard TTL circuits. See the Am9513 data sheet for specifications. In addition to providing TTL compatible voltage levels, other output conditions are specified to help configure non-standard interface circuitry. The logic level specifications take into account all worst-case combinations of the three variables that affect the logic level thresholds: ambient temperature, supply voltage and processing parameters. A change in any of these toward nominal values will improve the actual operating margins and will increase noise immunity.

Unprotected open gate inputs of high quality MOS transistors exhibit very high resistances on the order of perhaps $10^{14}$ ohms. It is easy, therefore, in some circumstances, for charge to enter the gate node of such an input faster than it can be discharged and consequently for the gate voltage to rise high enough to break down the oxides and destroy the transister. All inputs to the Am9513 include protection networks to help prevent damaging accumulations of static charge. The protection circuitry is designed to slow the transistions of incoming current surges and to provide low impedance discharge paths for voltages beyond the normal operating levels. Note, however, that input energy levels can nonetheless be too high to be successfully absorbed. Conventional design, storage, and handling precautions should be observed so that the protection networks themselves are not overstressed.

Within the limits of normal operation, the input protection circuitry is inactive and may be modeled as a lumped series RC as shown in Figure 1-7a. The functionality active input connection during normal operation is the gate of an MOS transistor. No active sources or drains are connected to the inputs so that neither transient nor steady-state currents are impressed on the driving signals other than the charging or discharging of the input capacitance and the accumulated leakage associated with the protection network and the input circuit.

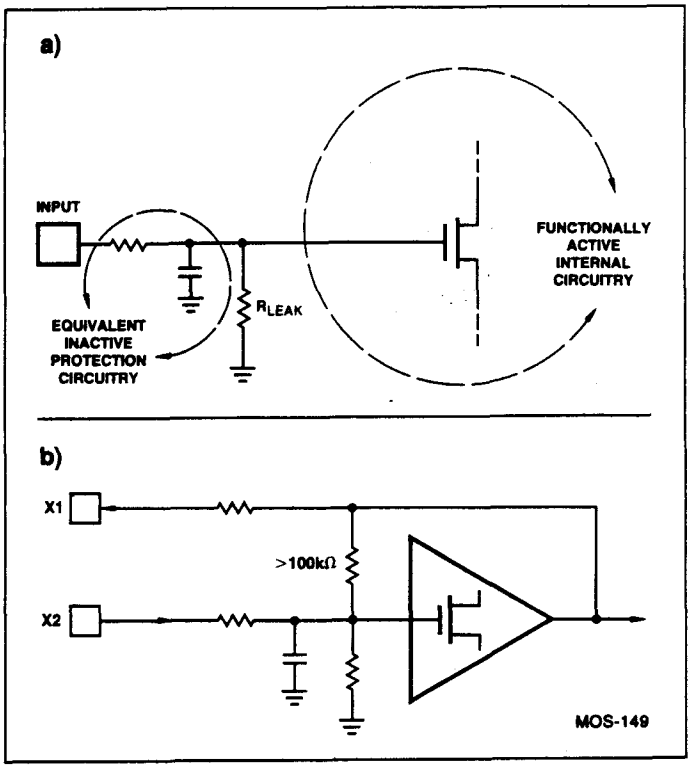



**Figure 1-7. Input Circuitry**

The only exception to the purely capacitive input case is the X2 crystal input. As shown in Figure 1-7b, an internal resistor connects X1 and X2 in addition to the protection network. The resistor is a modestly high value of more than 100kohms.

Fanout from the driving circuitry into the Am9513 inputs will generally be limited by transition time considerations rather than DC current limitations when the loading is dominated by conventional MOS circuits. In an operating environment, all inputs should be terminated so they do not float and therefore will not accumulate stray static charges. Unused inputs should be tied directly to Ground or VCC, as appropriate. An input in use will have some type of logic output driving it and termination during operation will not be a problem. Where inputs are driven from logic external to the card containing this chip, however, on-board termination should be provided to protect the chip when the board is unplugged and the input would therefore otherwise float. A pull-up resistor or a simple inverter or gate will suffice.

### Power Supply

The Am9513 requires only a single 5V power supply. Maximum supply currents are specified in the electrical specification at the high end of the voltage tolerance and the low end of the temperature range. In addition, the current specifications take into account the worstcase distribution of processing parameters that may be encountered during the manufacturing life of the product. Typical supply current values, on the other hand, are specified at a nominal +5.0 volts, a nominal ambient temperature of 25°C, and nominal processing parameters. Supply current always decreases with increasing ambient temperature: thermal run-away is not a problem.

Supply current will vary somewhat from part to part, but a given unit at a given operating temperature will exhibit a nearly constant power drain. There is no functional operating region that will cause more than a few percent change in the supply current. Decoupling of VCC, then, is straightforward and will generally be used to isolate the Am9513 from VCC noise originating externally.

## CONTROL PORT REGISTERS

The STC is addressed by the external system as only two locations: a Control port and a Data port. Transfers at the Control port (C/$\overline{D}$ = High) allow direct access to the command register when writing and the status register when reading. All other available internal locations are accessed for both reading and writing via the Data port (C/$\overline{D}$ = Low). Data port transfers are executed to and from the location currently addressed by the Data Pointer register. Options available in the Master Mode register and the Data Pointer control structure allow several types of transfer sequencing to be used. See Figure 1-8.

Transfers to and from the Control port are always 8-bits wide. Each access to the Control port will transfer data between the Command register (writes) or Status register (reads) and Data Bus pins DB0-DB7, regardless of whether the Am9513 is in 8- or 16-bit bus mode. When the Am9513 is in 8-bit bus mode, Data Bus pins DB13-DB15 should be held at a logic high whenever $\overline{CS}$ and WR are both active.

### Command Register

The Command register provides direct control over each of the five general counters and controls access through the Data port by allowing the user to update the Data Pointer register. The "Command Description" section of this data sheet explains the detailed operation of each command. A summary of all commands appears in Figure 1-21. Six of the command types are used for direct software control of the counting process. Each of these six commands contains a 5-bit S field. In a linear-select fashion, each bit in the S field corresponds to one of the five general counters (S1 = Counter 1, S2 = Counter 2, etc.). When an S bit is a one, the specified operation is performed on the counter so designated; when an S bit is a zero, no operation occurs for the corresponding counter.

### Data Pointer Register

The 6-bit Data Pointer register is loaded by issuing the appropriate command through the Control port to the Command register. As shown in Figure 1-8, the contents of the Data Pointer register are used to control the Data port multiplexer, selecting which internal register is to be accessible through the Data port.

The Data Pointer consists of a 3-bit Group Pointer, a 2-bit Element Pointer and a 1-bit Byte Pointer, depicted in Figure 1-9. The Byte Pointer bit indicates which byte of a 16-bit register is to be transferred on the next access through the Data port. Whenever the Data Pointer is loaded, the Byte Pointer bit is set to one, indicating a least-significant byte is expected. The Byte Pointer toggles following each 8-bit data transfer with an 8-bit data bus (MM13 = 0), or it always remains set with the 16-bit data bus option (MM13 = 1). The Element and Group pointers are used to select which internal register is to be accessible through the Data port. Although the contents of the Element and Group Pointer in the Data Pointer register cannot be read by the host processor, the Byte Pointer is available as a bit in the Status register.

Random access to any available internal data location can be accomplished by simply loading the Data Pointer using the command shown in Figure 1-10 and then initiating a data read or data write. This procedure can be used at any time, regardless of the setting of the Data Pointer Control bit (MM14). When the 8-bit data bus configuration is being used (MM13 = 0), two bytes of data would normally be transferred following the issuing of the "Load Data Pointer" command.

To permit the host processor to rapidly access the various internal registers, automatic sequencing of the Data Pointer is provided.

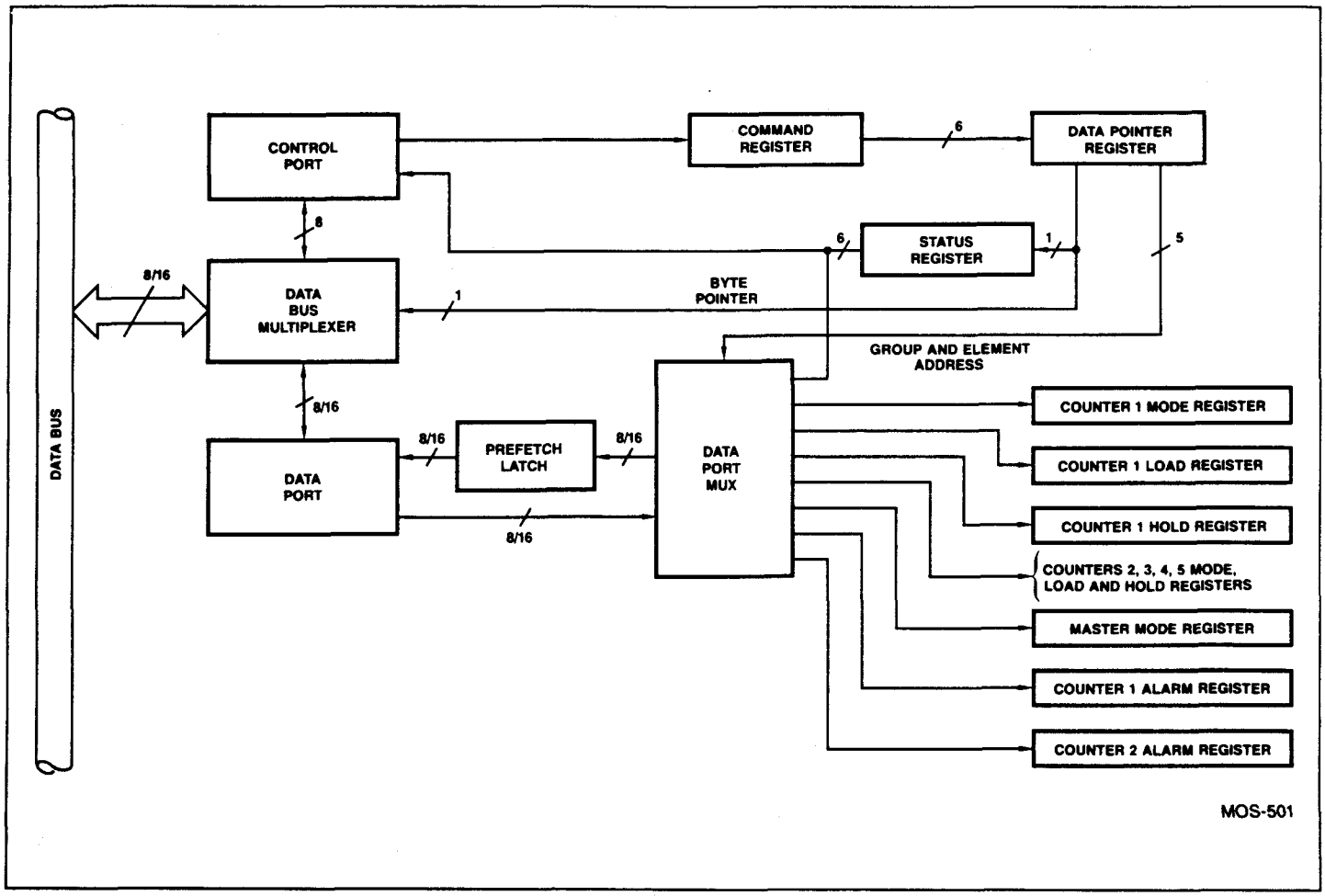



**Figure 1-8. Am9513 Register Access**

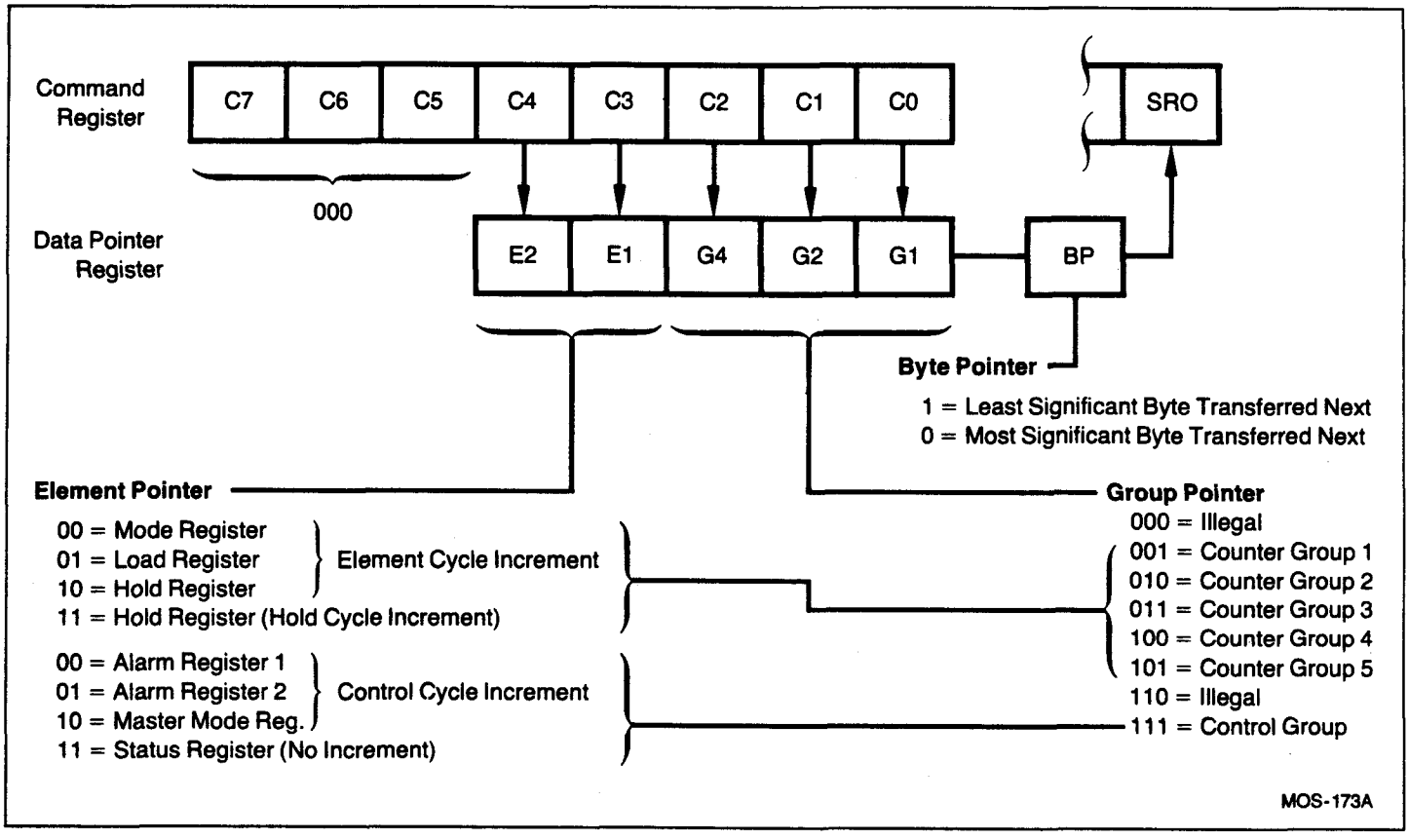



**Figure 1-9. Data Pointer Register**

| | Element Cycle | | | Hold Cycle |
|---|---|---|---|---|
| | Mode Register | Load Register | Hold Register | Hold Register |
| Counter 1 | FF01 | FF09 | FF11 | FF19 |
| Counter 2 | FF02 | FF0A | FF12 | FF1A |
| Counter 3 | FF03 | FF0B | FF13 | FF1B |
| Counter 4 | FF04 | FF0C | FF14 | FF1C |
| Counter 5 | FF05 | FF0D | FF15 | FF1D |

Master Mode Register = FF17
Alarm 1 Register = FF07
Alarm 2 Register = FF0F
Status Register = FF1F

Notes:
1. All codes are in hex.
2. When used with an 8-bit bus, only the two low order hex digits should be written to the command port; the 'FF' prefix should be used only for a 16-bit data bus interface.

**Figure 1-10. Load Data Pointer Commands**

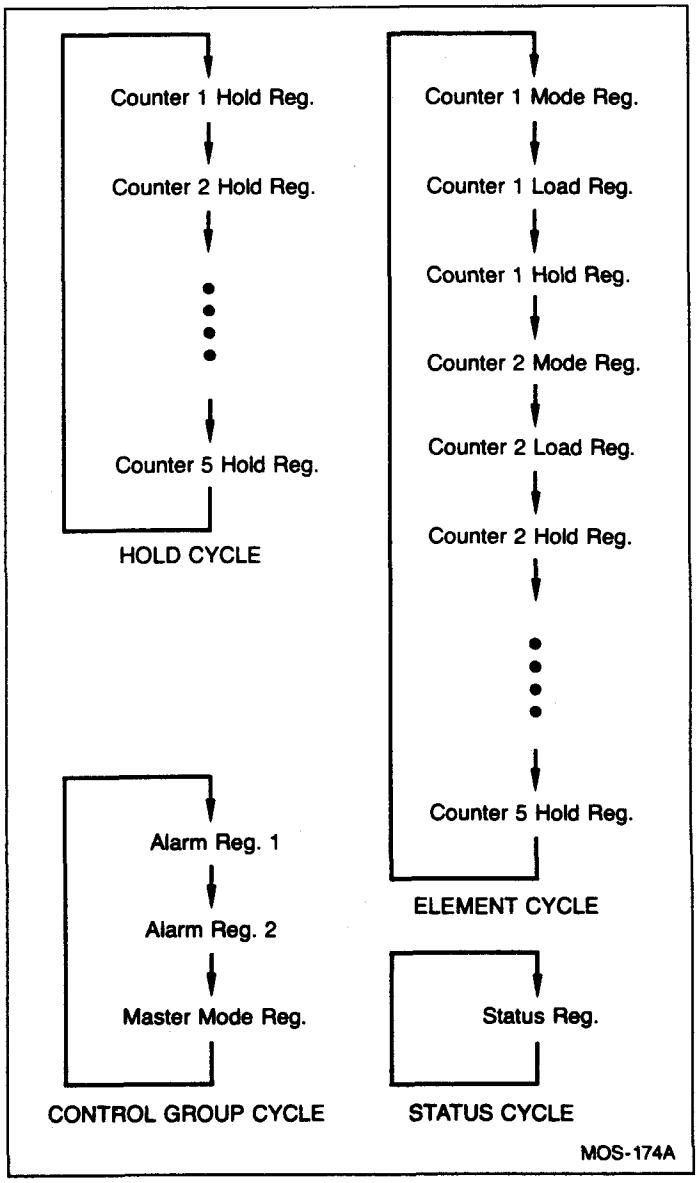



**Figure 1-11. Data Pointer Sequencing**

Sequencing is enabled by clearing Master Mode bit 14 (MM14) to zero. As shown in Figure 1-11, several types of sequencing are available depending on the data bus width being used and the initial Data Pointer value entered by command.

When E1 = 0 or E2 = 0 and G4, G2, G1 point to a Counter Group, the Data Pointer will proceed through the Element cycle. The Element field will automatically sequence through the three values 00, 01 and 10 starting with the value entered. When the transition from 10 to 00 occurs, the Group field will also be incremented by one. Note that the Element field in this case does not sequence to a value of 11. The Group field circulates only within the five Counter Group codes.

If E2, E1 = 11 and a Counter Group is selected, then only the Group field is sequenced. This is the Hold cycle. It allows the Hold registers to be sequentially accessed while bypassing the Mode and Load registers. The third type of sequencing is the Control cycle. If G4, G2, G1 = 111 and E2, E1 $\neq$ 11, the Element Pointer will be incremented through the values 00, 01 and 10, with no change to the Group Pointer.

When G4, G2, G1 = 111 and E2, E1 = 11, no incrementing takes place and only the Status register will be available through the Data port. Note that the Status register can also always be read directly through the Control port.

For all of these auto-sequence modes, if an 8-bit data bus is used, the Byte pointer will toggle after every data transfer to allow the least and most significant bytes to be transferred before the Element or Group Fields are incremented.

### Prefetch Circuit

In order to minimize the read access time to internal Am9513 registers, a prefetch circuit is used for all read operations through the Data port. Following each read or write operation through the Data port, the Data Pointer register is updated to point to the next register to be accessed. Immediately following this update, the new register data is transferred to a special prefetch latch at the interface pad logic. When the user performs a subsequent read of the Data port, the data bus drivers are enabled, outputting the prefetched data on the bus. Since the internal data register is accessed prior to the start of the read operation, its access time is transparent to the user. In order to keep the prefetched data consistent with the Data Pointer, prefetches are also performed after each write to the Data port and after execution of the "Load Data Pointer" command. The following rules should be kept in mind regarding Data port Transfers.

1. The Data Pointer register should always be reloaded before reading from the Data port if a command other than "Load Data Pointer" was issued to the Am9513 following the last Data port read or write. The Data Pointer does not have to be loaded again if the first Data port transaction after a command entry is a write, since the Data port write will automatically cause a new prefetch to occur.
2. Operating modes N, O, Q, R and X allow the user to save the counter contents in the Hold register by applying an active-going gate edge. If the Data Pointer register had been pointing to the Hold register in question, the prefetched value will not correspond to the new value saved in the Hold register. To avoid reading an incorrect value, a new "Load Data Pointer" command should be issued before attempting to read the saved data. A Data port write (to another register) will also initiate a prefetch; subsequent reads will access the recently saved Hold register data. Many systems will use the "saving" gate edge to interrupt the host CPU. In systems such as this the interrupt service routine should issue a "Load Data Pointer" command prior to reading the saved data.

### Status Register

The 8-bit read-only Status register indicates the state of the Byte Pointer bit in the Data Pointer register and the state of the OUT signal for each of the general counters. See Figures 1-12 and 1-19. The OUT signals reported are those internal to the chip after the polarity-select logic and just before the 3-state interface buffer circuitry. Bits SR6 and SR7 may be 0 or 1.

The Status register OUT bit reflects an active-high or active-low TC output, or a TC Toggled output, as programmed in the Output Control Field of the Counter Mode register. That is, it reflects the exact state of the OUT pin. When the Low Impedance to Ground Output option (CM2-CM0 = 000) is selected, the Status register will reflect an active-high TC Output. When a High Impedance Output option (CM2-CM0 = 100) is selected, the Status register will reflect an active-low TC output.

For Counters 1 and 2, the OUT pin will reflect the comparator output if the comparators are enabled. The Status register bit and OUT pin are active high if CM2 = 0 and active-low if CM2 = 1. When the High Impedance option is selected and the comparator is enabled, the status register bit will reflect an active-high comparator output. When the Low Impedance to Ground option is selected and the comparator is enabled, the status register bit will reflect an active-low comparator output.

The Status register is normally accessed by reading the Control port (see Figure 1-8) but may also be read via the Data port as part of the Control Group.

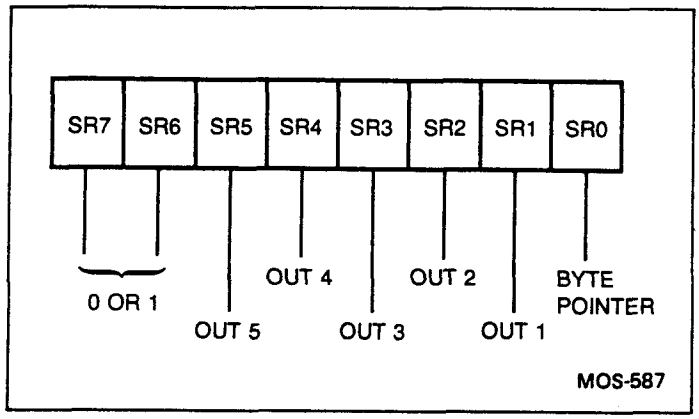



**Figure 1-12. Status Register Bit Assignments**

## DATA PORT REGISTERS

### Counter Logic Groups

As shown in Figures 1-2 and 1-3, each of the five Counter Logic Groups consists of a 16-bit general counter with associated control and output logic, a 16-bit Load register, a 16-bit Hold register and a 16-bit Mode register. In addition, Counter Groups 1 and 2 also include 16-bit Comparators and 16-bit Alarm registers. The comparator/alarm functions are controlled by the Master Mode register. The operation of the Counter Mode registers is the same for all five counters. The host CPU has both read and write access to all registers in the Counter Logic Groups through the Data port. The counter itself is never directly accessed.

### Load Register

The 16-bit read/write Load register is used to control the effective length of the general counter. Any 16-bit value may be written into the Load register. That value can then be transferred into the counter each time the Terminal Count (TC) occurs. "Terminal Count" is defined as that period of time when the counter contents would have been zero if an external value had not been transferred into the counter. Thus, the terminal count frequency can be the input frequency divided by the value in the Load register. In all operating modes either the Load or Hold register will be transferred into the counter when TC occurs. In cases where values are being accumulated in the counter, the Load register action can become transparent by filling the Load register with all zeros.

### Hold Register

The 16-bit read/write Hold register is dual-purpose. It can be used in the same way as the Load register, thus offering an alternate source for module definition for the counter. The Hold register may also be used to store accumulated counter values for later transfer to the host processor. This allows the count to be sampled while the counting process proceeds without interruption. Transfer of the counter contents into the Hold register is accomplished by the hardware interface in some operating modes or by software commands at any time.

### Counter Mode Register

The 16-bit read/write Counter Mode register controls the gating, counting, output and source select functions within each Counter Logic Group. The "Counter Mode Control Options" section of this document describes the detailed control options available. Figure 1-18 shows the bit assignments for the Counter Mode registers.

### Alarm Registers and Comparators

Added functions are available in the Counter Logic Groups for Counters 1 and 2 (see Figure 1-2). Each contains a 16-bit Alarm register and a 16-bit Comparator. When the value in the counter reaches the value in the Alarm register, the Comparator output will go true. The Master Mode register contains control bits to individually enable/disable the comparators. When enabled, the comparator output appears on the OUT pin of the associated counter in place of the normal counter output. The output will remain true as long as the comparison is true, that is, until the next input causes the count to change. The polarity of the Comparator output will be active-high if the Output Control field of the Counter Mode register is 001 or 010 and active-low if the Output Control field is 101.

## MASTER MODE CONTROL OPTIONS

The 16-bit Master Mode (MM) register is used to control those internal activities that are not controlled by the individual Counter Mode registers. This includes frequency control, Time-of-Day operation, comparator controls, data bus width and data pointer sequencing. Figure 1-13 shows the bit assignments for the Master Mode register. This section describes the use of each control field.

Master Mode register bits MM12, MM13 and MM14 can be individually set and reset using commands issued to the Command register. In addition they can all be changed by writing directly to the Master Mode register.

After power-on reset or a Master Reset command, the Master Mode register is cleared to an all zero condition. This results in the following configuration:

Time-of-Day disabled
Both Comparators disabled
FOUT Source is frequency F1
FOUT Divider set for divide-by-16
FOUT gated on
Data Bus 8 bits wide
Data Pointer Sequencing enabled
Frequency Scaler divides in binary

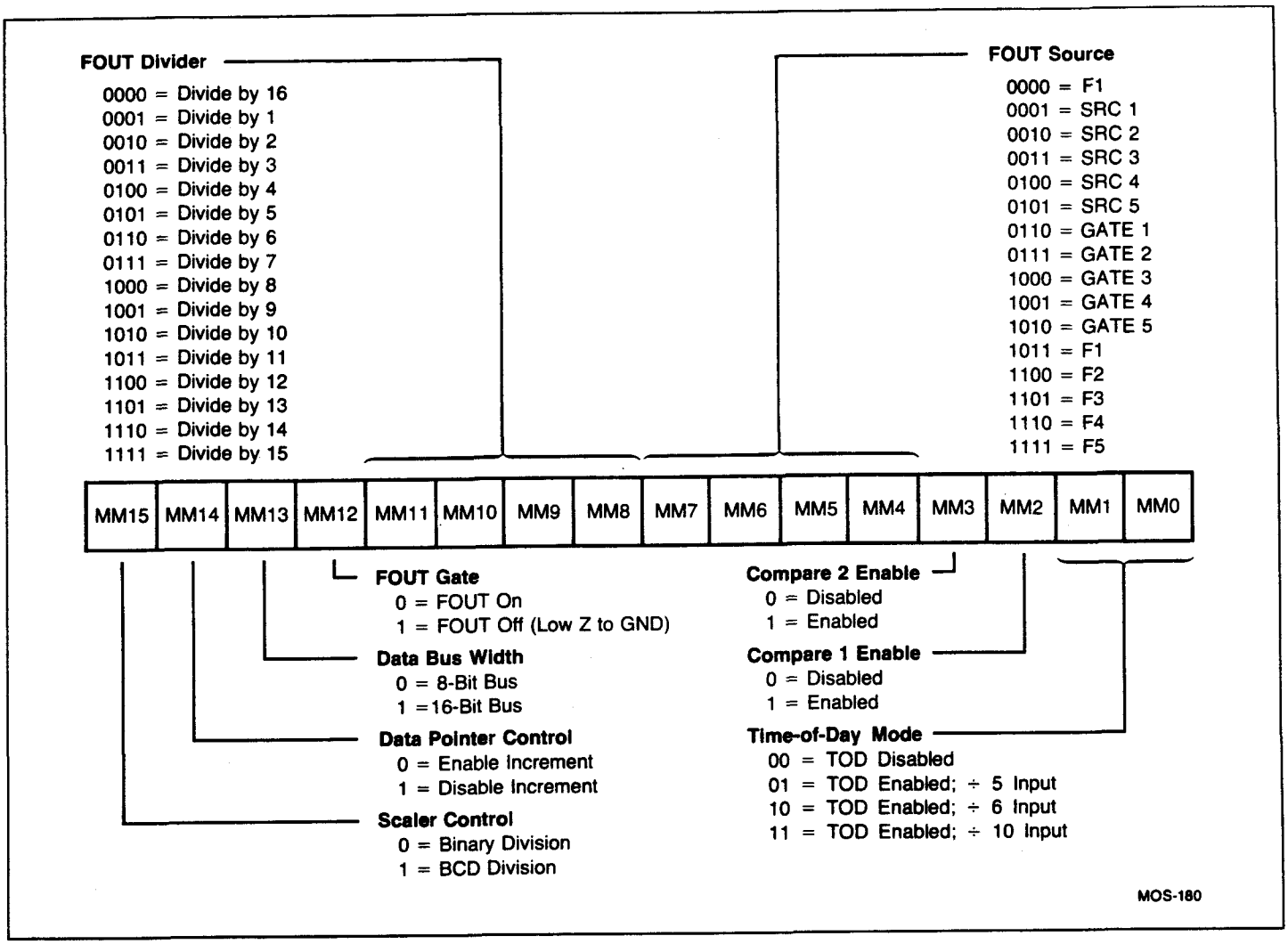



**Figure 1-13. Master Mode Register Bit Assignments**

### Time-of-Day

Bits MM0 and MM1 of the Master Mode register specify the Time-of-Day (TOD) options. When MM0 = 0 and MM1 = 0, the special logic used to implement TOD is disabled and Counters 1 and 2 will operate in exactly the same way as Counters 3, 4 and 5. When MM0 = 1 or MM1 = 1, additional counter decoding and control logic is enabled on Counters 1 and 2 which causes their decades to turn over at the counts that generate appropriate 24-hour TOD accumulations. For additional information, see the Time-of-Day chapter in this applications note.

### Comparator Enable

Bits MM2 and MM3 control the Comparators associated with Counter 1 and 2. When a Comparator is enabled, its output is substituted for the normal counter output on the associated OUT1 or OUT2 pin. The comparator output will be active-high if the output control field of the Counter Mode register is 001 or 010 and active low for a code of 101. Once the compare output is true, it will remain so until the count changes and the comparison therefore goes false.

The two Comparators can always be used individually in any operating mode. One special case occurs when the Time-of-Day option is invoked and both Comparators are enabled. The operation of Comparator 2 will then be conditioned by Comparator 1 so that a full 32-bit compare must be true in order to generate a true signal on OUT2. OUT1 will continue, as usual, to reflect the state of the 16-bit comparison between Alarm 1 and Counter 1.

### FOUT Source

Master Mode bits MM4 through MM7 specify the source input for the FOUT divider. Fifteen inputs are available for selection and they include the five Source pins, the five Gate pins and the five internal frequencies derived from the oscillator. The 16th combination of the four control bits (all zeros) is used to assure that an active frequency is available at the input to the FOUT divider following reset.

### FOUT Divider

Bits MM8 through MM11 specify the dividing ratio for the FOUT Divider. The FOUT source (selected by bits MM4 through MM7) is divided by an integer value between 1 and 16, inclusive, and is then passed to the FOUT output buffer. After power-on or reset, the FOUT divider is set to divide-by-16.

### FOUT Gate

Master Mode bit MM12 provides a software gating capability for the FOUT signal. When MM12 = 1, FOUT is off and in a low impedance state to ground. MM12 may be set or cleared in conjunction with the loading of the other bits in the Master Mode register; alternatively, there are commands that allow MM12 to be individually set or cleared directly without changing any other Master Mode bits. After power-up or reset, FOUT is gated on.

When changing the FOUT divider ratio or FOUT source, transient pulses as short as half the period of the FOUT source may appear

on the FOUT pin. Turning the FOUT gate on or off can also generate a transient. This should be considered when using FOUT as a system clock source.

### Bus Width

Bit MM13 controls the multiplexer at the data bus interface in order to configure the part for an 8-bit or 16-bit external bus. The internal bus is always 16-bits wide. When MM13 = 1, 16-bit data is transferred directly between the internal bus and all 16 of the external bus lines. In this configuration, the Byte Pointer bit in the Data Pointer register remains set at all times. When MM13 = 0, 16-bit internal data is transferred a byte at a time to and from the eight low-order external data bus lines. The Byte Pointer bit toggles with each byte transfer in this mode.

When the Am9513 is set to operate with an 8-bit data bus width, pins DB8 through DB15 are not used for the data bus and are available for other functions. Pins DB13 through DB15 should be tied high. Pins DB8 through DB12 are used as auxiliary gating inputs, and are labeled GATE1A through GATE5A respectively. The auxiliary gate pin, GATENA, is logically ANDed with the gate input to Counter N, as shown in Figure 1-14. The output of the AND gate is then used as the gating signal for Counter N.

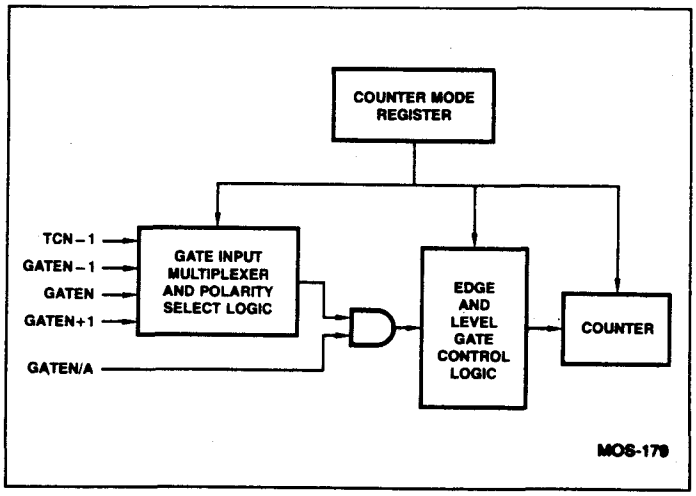



**Figure 1-14. Gating Control**

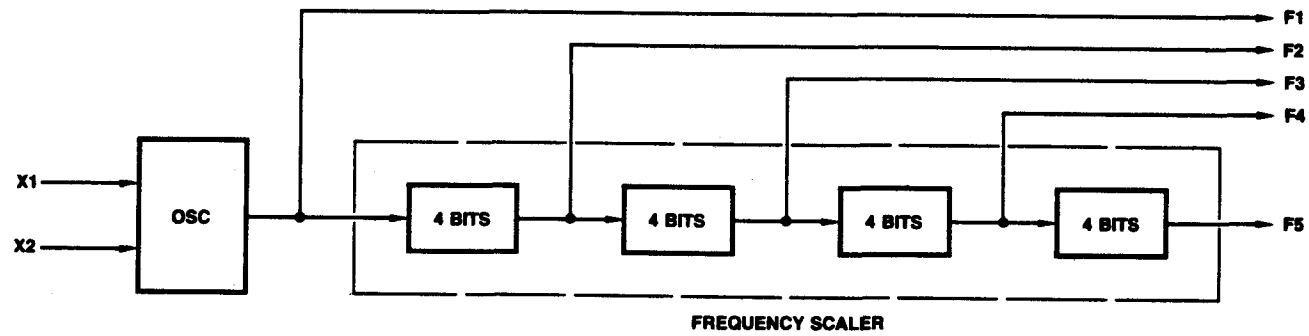
### Data Pointer Sequencing

Bit MM14 controls the Data Pointer logic to enable or disable the automatic sequencing functions. When MM14 = 1, the contents of the Data Pointer can be changed only directly by entering a command. When MM14 = 0, several types of automatic sequencing of the Data Pointer are available. These are described in the Data Pointer register section of this document.

Thus the host processor, by controlling MM14, may repetitively read/write a single internal location, or may sequentially read/write groups of locations. Bit MM14 can be loaded by writing to the Master Mode register or can be set or cleared by software command.

### Scaler Ratios

Master Mode bit MM15 controls the counting configuration of the Frequency Scaler counter. When MM15 = 0, the Scaler divides the oscillator frequency in binary steps so that each subfrequency is 1/16 of the preceding frequency. When MM15 = 1, the Scaler divides in BCD steps so that adjacent frequencies are related by ratios of 10 instead of 16 (see Figure 1-15).




| Frequency | BCD Scaling MM15 = 1 | Binary Scaling MM15 = 0 |
|---|---|---|
| F1 | OSC | OSC |
| F2 | F1 ÷ 10 | F1 ÷ 16 |
| F3 | F1 ÷ 100 | F1 ÷ 256 |
| F4 | F1 ÷ 1,000 | F1 ÷ 4,096 |
| F5 | F1 ÷ 10,000 | F1 ÷ 65,536 |

MOS-150

**Figure 1-15. Frequency Scaler Ratios**

| Counter Mode | A | B | C | D | E | F | G | H | I | J | K | L |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Special Gate (CM7) | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Reload Source (CM6) | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| Repetition (CM5) | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| Gate Control (CM15-CM13) | 000 | LEVEL | EDGE | 000 | LEVEL | EDGE | 000 | LEVEL | EDGE | 000 | LEVEL | EDGE |
| Count to TC once, then disarm | X | X | X | | | | | | | | | |
| Count to TC twice, then disarm | | | | | | | X | X | X | | | |
| Count to TC repeatedly without disarming | | | | X | X | X | | | | X | X | X |
| Gate input does not gate counter input | X | | | X | | | X | | | X | | |
| Count only during active gate level | | X | | | X | | | X | | | X | |
| Start count on active gate edge and stop count on next TC | | | X | | | X | | | | | | |
| Start count on active gate edge and stop count on second TC | | | | | | | | | X | | | X |
| No hardware retriggering | X | X | X | X | X | X | X | X | X | X | X | X |
| Reload counter from Load Register on TC | X | X | X | X | X | X | | | | | | |
| Reload counter on each TC, alternating reload source between Load and Hold Registers | | | | | | | X | X | X | X | X | X |
| Transfer Load Register into counter on each TC that gate is LOW, transfer Hold Register into counter on each TC that gate is HIGH. | | | | | | | | | | | | |
| On active gate edge transfer counter into Hold Register and then reload counter from Load Register | | | | | | | | | | | | |

| Counter Mode | M | N | O | P | Q | R | S | T | U | V | W | X |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Special Gate (CM7) | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Reload Source (CM6) | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| Repetition (CM5) | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| Gate Control (CM15-CM13) | 000 | LEVEL | EDGE | 000 | LEVEL | EDGE | 000 | LEVEL | EDGE | 000 | LEVEL | EDGE |
| Count to TC once, then disarm | | X | X | | | | | | | | | |
| Count to TC twice, then disarm | | | | | | | X | | | | | |
| Count to TC repeatedly without disarming | | | | | X | X | | | | X | | X |
| Gate input does not gate counter input | | | | | | | X | | | X | | |
| Count only during active gate level | | X | | | X | | | | | | | |
| Start count on active gate edge and stop count on next TC | | | X | | | X | | | | | | X |
| Start count on active gate edge and stop count on second TC | | | | | | | | | | | | |
| No hardware retriggering | | | | | | | X | | | X | | X |
| Reload counter from Load Register on TC | | X | X | | X | X | | | | | | X |
| Reload counter on each TC, alternating reload source between Load and Hold Registers. | | | | | | | | | | | | |
| Transfer Load Register into counter on each TC that gate is LOW, transfer Hold Register into counter on each TC that gate is HIGH. | | | | | | | X | | | X | | |
| On active gate edge transfer counter into Hold Register and then reload counter from Load Register | | X | X | | X | X | | | | | | |
| On active gate edge transfer counter into Hold Register, but counting continues | | | | | | | | | | | | X |

Notes:
1. Counter modes M, P, T, U and W are reserved and should not be used.
2. Mode X is available for Am9513A only.

**Figure 1-16. Counter Mode Operating Summary**

## COUNTER MODE DESCRIPTIONS

Counter Mode register bits CM15-CM13 and CM7-CM5 select the operating mode for each counter (see Figure 1-16). To simplify references to a particular mode, each mode is assigned a letter from A through X. Representative waveforms for the counter modes are illustrated in Figures 1-17a through 1-17v. (Because the letter suffix in the figure number is keyed to the mode, Figures 1-17m, 1-17p, 1-17t, 1-17u and 1-17w do not exist.) The figures assume down counting on rising source edges. Those modes which automatically disarm the counter (CM5 = 0) are shown with the $\overline{WR}$ pulse entering the required ARM command; for modes which count repetitively (CM5 = 1) the ARM command is omitted. The retriggering modes (N, O, Q and R) are shown with one retrigger operation. Both a TC output waveform and a TC Toggled output waveform are shown for each mode. The symbols L and H are used to represent count values equal to the Load and Hold register contents, respectively. The symbols K and N represent arbitrary count values. For each mode, the required bit pattern in the Counter Mode register is shown; "don't care" bits are marked "X." These figures are designed to clarify the mode descriptions; the Am9513 Electrical Specification should be used as the authoritative reference for timing relationships between signals. Appendix B provides a key to the waveform symbols used in these diagrams.

To keep the following mode descriptions concise and to the point, the phrase "source edges" is used to refer to active-going source edges only, not to inactive-going edges. Similarly, the phrase "gate edges" refers only to active-going gate edges. Also, again to avoid verbosity and euphuism, the descriptions of some modes state that a counter is stopped or disarmed "on a TC, inhibiting further counting." As is fully explained in the TC section of this document, for these modes the counter is actually stopped or disarmed following the active-going source edge which drives the counter out of TC. In other words, since a counter in the TC state always counts, irrespective of its gating or arming status, the stopping or disarming of the count sequence is delayed until TC is terminated.

### MODE A

**Software-Triggered Strobe with No Hardware Gating**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | X | X |

Mode A, shown in Figure 1-17a, is one of the simplest operating modes. The counter will be available for counting source edges when it is issued an ARM command. On each TC the counter will reload from the Load register and automatically disarm itself, inhibiting further counting. Counting will resume when a new ARM command is issued.

### MODE B

**Software-Triggered Strobe with Level Gating**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| LEVEL | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | X | X |

Mode B, shown in Figure 1-17b, is identical to Mode A except that source edges are counted only when the assigned Gate is active. The counter must be armed before counting can occur. Once armed, the counter will count all source edges which occur while the Gate is active and disregard those edges which occur while the Gate is inactive. This permits the Gate to turn the count process on and off. On each TC the counter will reload from the Load register and automatically disarm itself, inhibiting further counting until a new ARM command is issued.

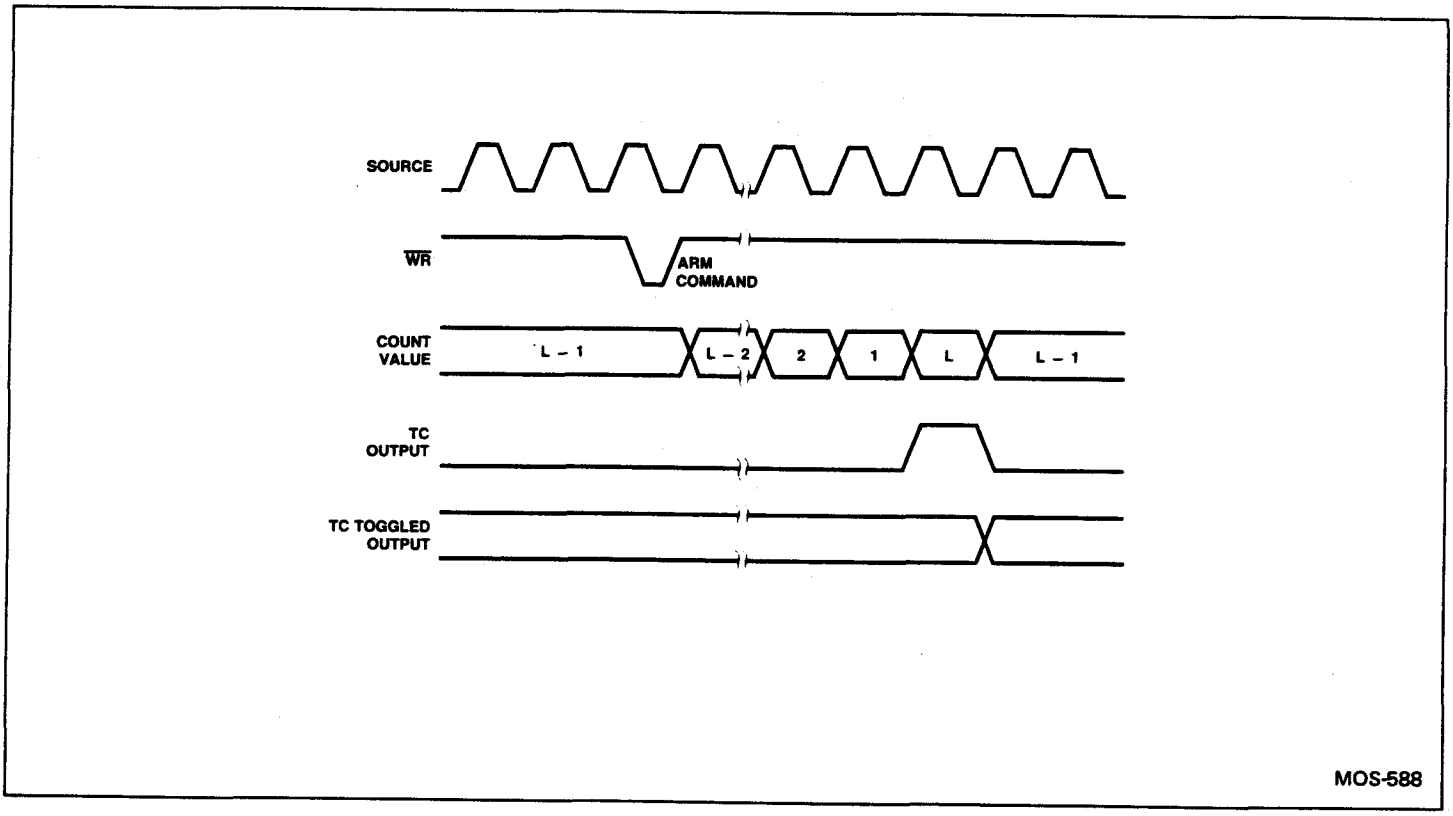



**Figure 1-17a. Mode A Waveforms**

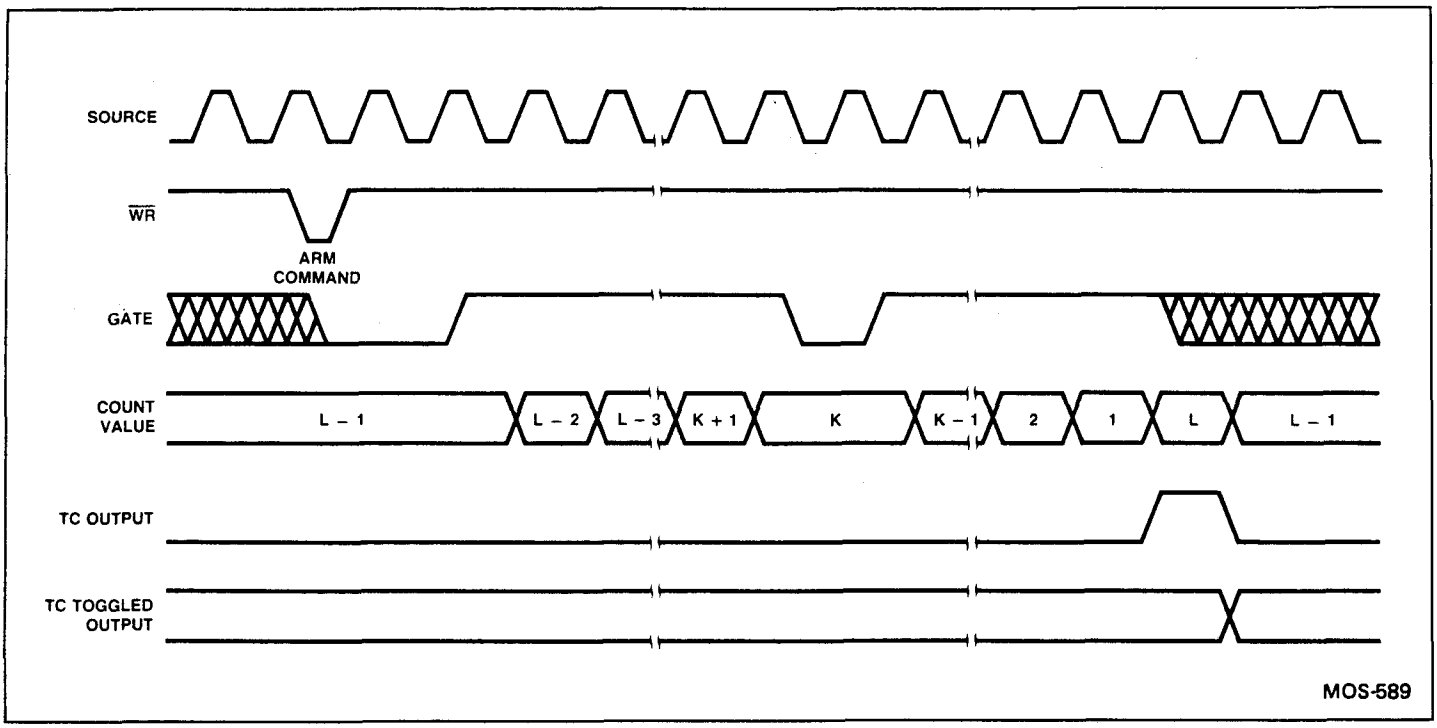



**Figure 1-17b. Mode B Waveforms**

## MODE C
### Hardware-Triggered Strobe

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| EDGE | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | X | X |

Mode C, shown in Figure 1-17c, is identical to Mode A, except that counting will not begin until a Gate edge is applied to the armed counter. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. The counter will start counting on the first source edge after the triggering Gate edge and will continue counting until TC. At TC, the counter will reload from the Load register and automatically disarm itself. Counting will then remain inhibited until a new ARM command and a new Gate edge are applied in that order. Note that after application of a triggering Gate edge, the Gate input will be disregarded for the remainder of the count cycle. This differs from Mode B, where the Gate can be modulated throughout the count cycle to stop and start the counter.

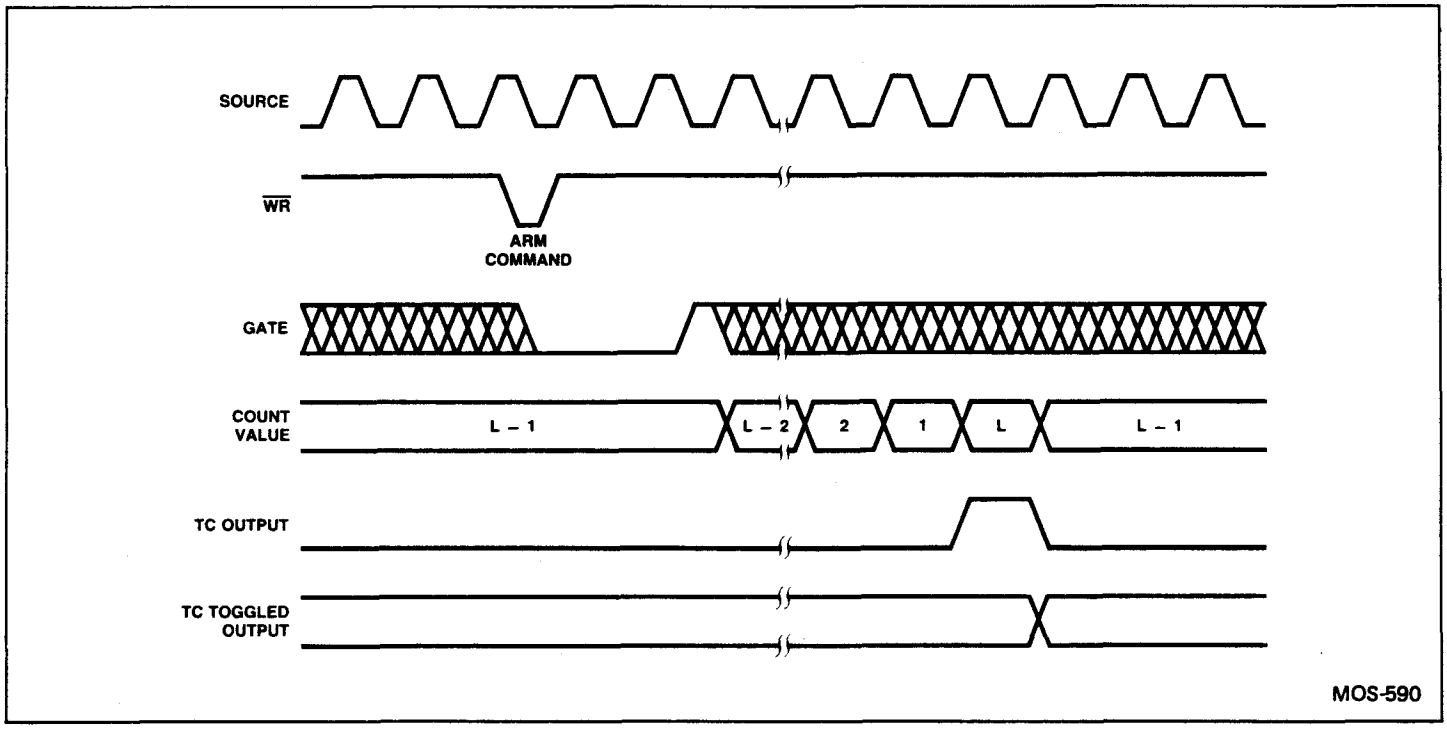



**Figure 1-17c. Mode C Waveforms**

**MODE D**
**Rate Generator with No Hardware Gating**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | X | X | X | X | X |

Mode D, shown in Figure 1-17d, is typically used in frequency generation applications. In this mode, the Gate input does not affect counter operation. Once armed, the counter will count to TC repetitively. On each TC the counter will reload itself from the Load register; hence the Load register value determines the time between TCs. A square wave rate generator may be obtained by specifying the TC Toggled output mode in the Counter Mode register.

**MODE E**
**Rate Generator with Level Gating**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| LEVEL | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | X | X | X | X | X |

Mode E, shown in Figure 1-17e, is identical to Mode D, except the counter will only count those source edges which occur while the Gate input is active. This feature allows the counting process to be enabled and disabled under hardware control. A square wave rate generator may be obtained by specifying the TC Toggled output mode.

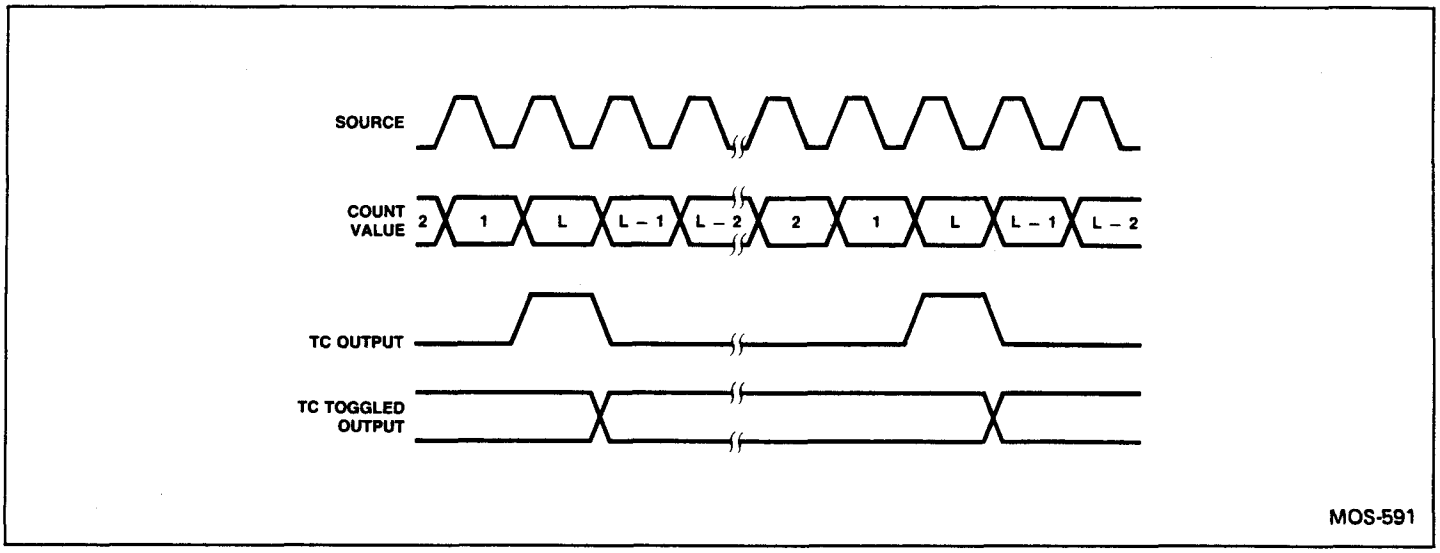



**Figure 1-17d. Mode D Waveforms**

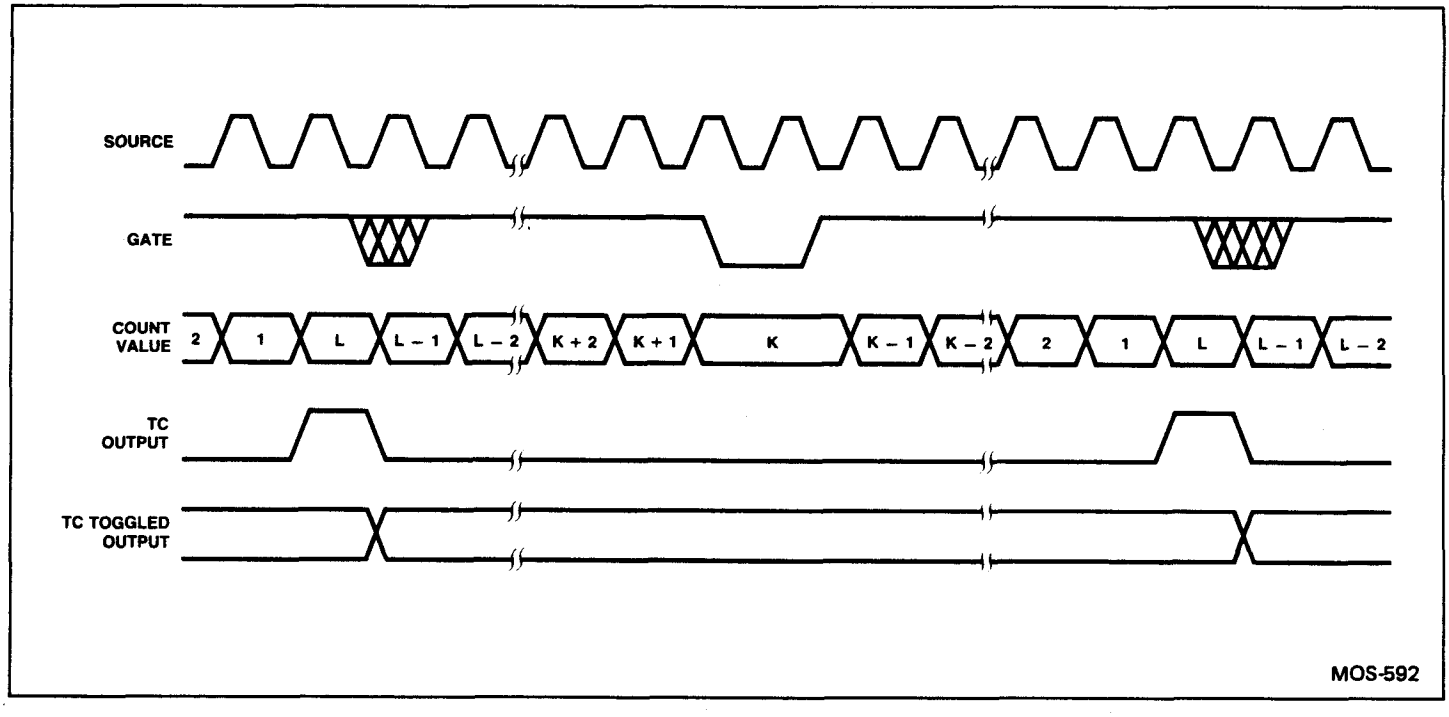



**Figure 1-17e. Mode E Waveforms**

**MODE F**
**Non-Retriggerable One-Shot**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| EDGE | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | X | X | X | X | X |

Mode F, shown in Figure 1-17f, provides a non-retriggerable one-shot timing function. The counter must be armed before it will function. Application of a Gate edge to the armed counter will enable counting. When the counter reaches TC, it will reload itself from the Load register. The counter will then stop counting, awaiting a new Gate edge. Note that unlike Mode C, a new ARM command is not needed after TC, only a new Gate edge. After application of a triggering Gate edge, the Gate input is disregarded until TC.

**MODE G**
**Software-Triggered Delayed Pulse One-Shot**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | X | X | X | X | X |

In Mode G, the Gate does not affect the counter's operation. Once armed, the counter will count to TC twice and then automatically disarm itself. For most applications, the counter will initially be loaded from the Load register either by a LOAD command or by the last TC of an earlier timing cycle. Upon counting to the first TC, the counter will reload itself from the Hold register. Counting will proceed until the second TC, when the counter will reload itself from the Load register and automatically disarm itself, inhibiting further counting. Counting can be resumed by issuing a new ARM command. A software-triggered delayed pulse one-shot may be generated by specifying the TC Toggled output mode in the Counter Mode register. The initial counter contents control the delay from the ARM command until the output pulse starts. The Hold register contents control the pulse duration. Mode G is shown in Figure 1-17g.

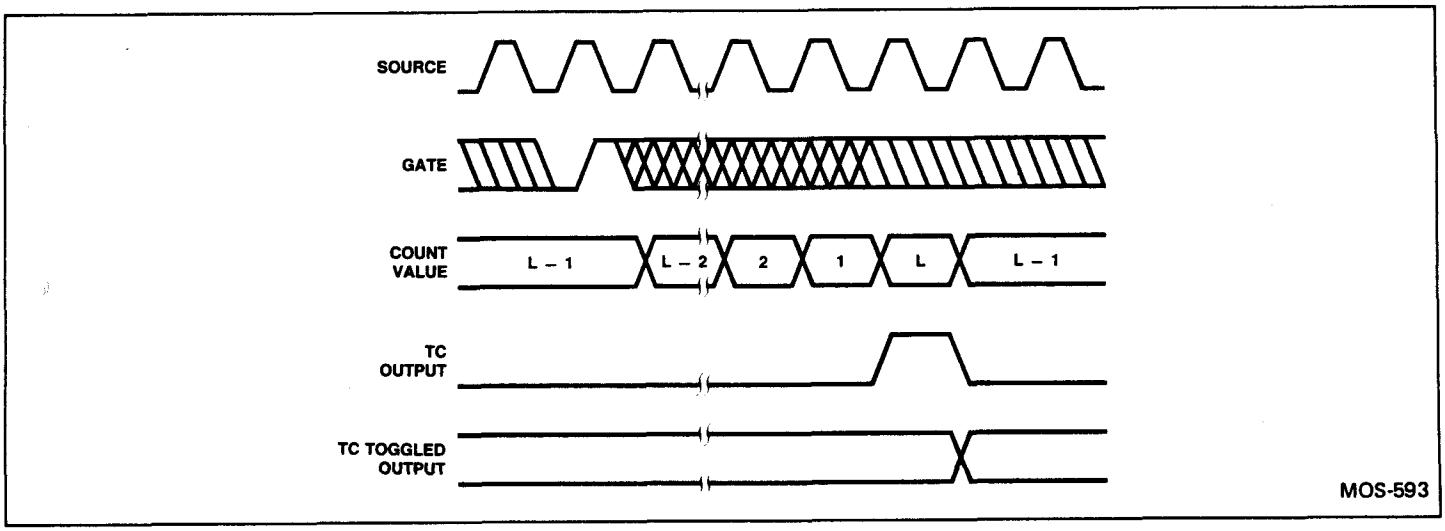



**Figure 1-17f. Mode F Waveforms**

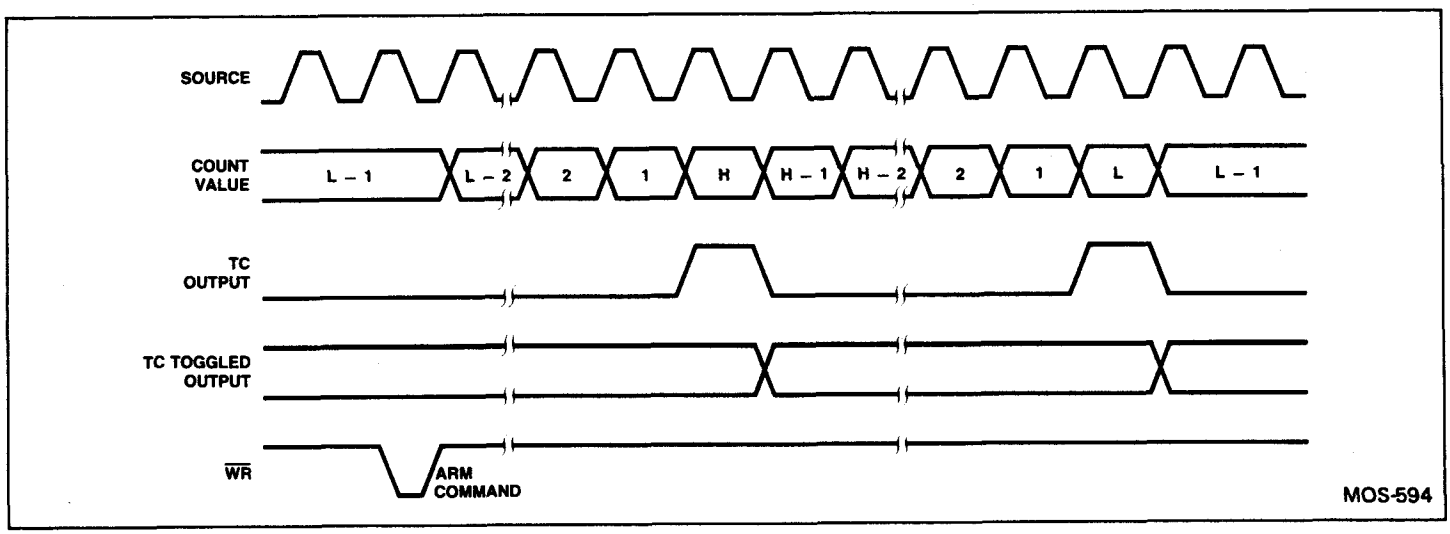



**Figure 1-17g. Mode G Waveforms**

**MODE H**
**Software-Triggered Delayed Pulse One-Shot with Hardware Gating**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| LEVEL | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | X | X | X | X | X |

Mode H, shown in Figure 1-17h, is identical to Mode G except that the Gate input is used to qualify which source edges are to be counted. The counter must be armed for counting to occur. Once armed, the counter will count all source edges that occur while the Gate is active and disregard those source edges that occur while the Gate is inactive. This permits the Gate to turn the count process on and off. As with Mode G, the counter will be reloaded from the Hold register on the first TC and reloaded from the Load register and disarmed on the second TC. This mode allows the Gate to control the extension of both the initial output delay time and the pulse width.

**MODE I**
**Hardware-Triggered Delayed Pulse Strobe**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| EDGE | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | X | X | X | X | X |

Mode I, shown in Figure 1-17i, is identical to Mode G, except that counting will not begin until a Gate edge is applied to an armed counter. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. An armed counter will start counting on the first source edge after the triggering Gate edge. Counting will then proceed in the same manner as in Mode G. After the second TC, the counter will disarm itself. An ARM command and Gate edge must be issued in this order to restart counting. Note that after application of a triggering Gate edge, the Gate input will be disregarded until the second TC. This differs from Mode H, where the Gate can be modulated throughout the count cycle to stop and start the counter.

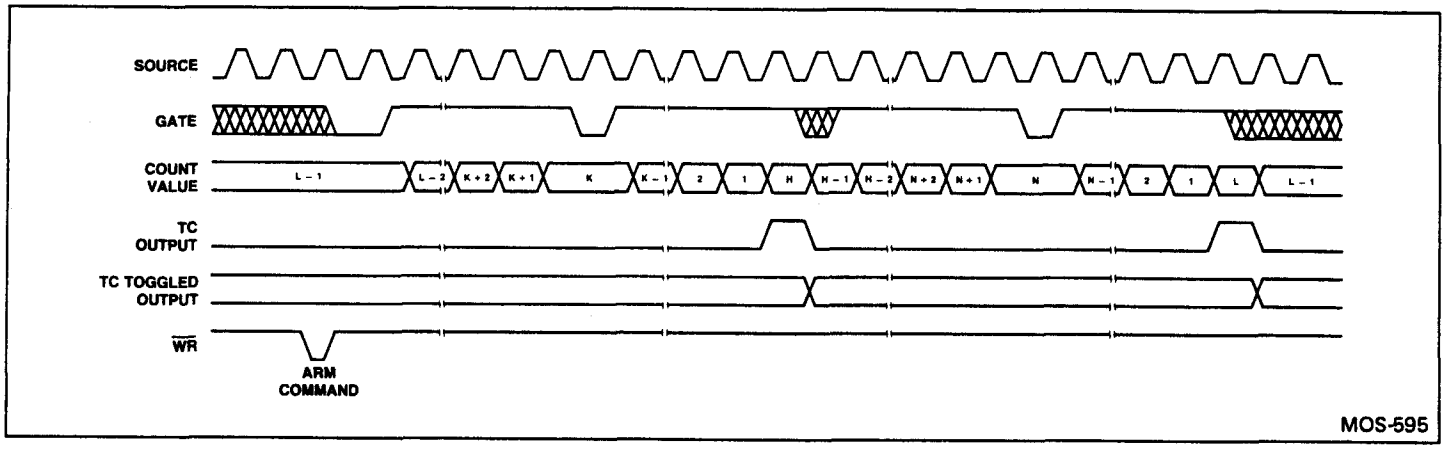



**Figure 1-17h. Mode H Waveforms**

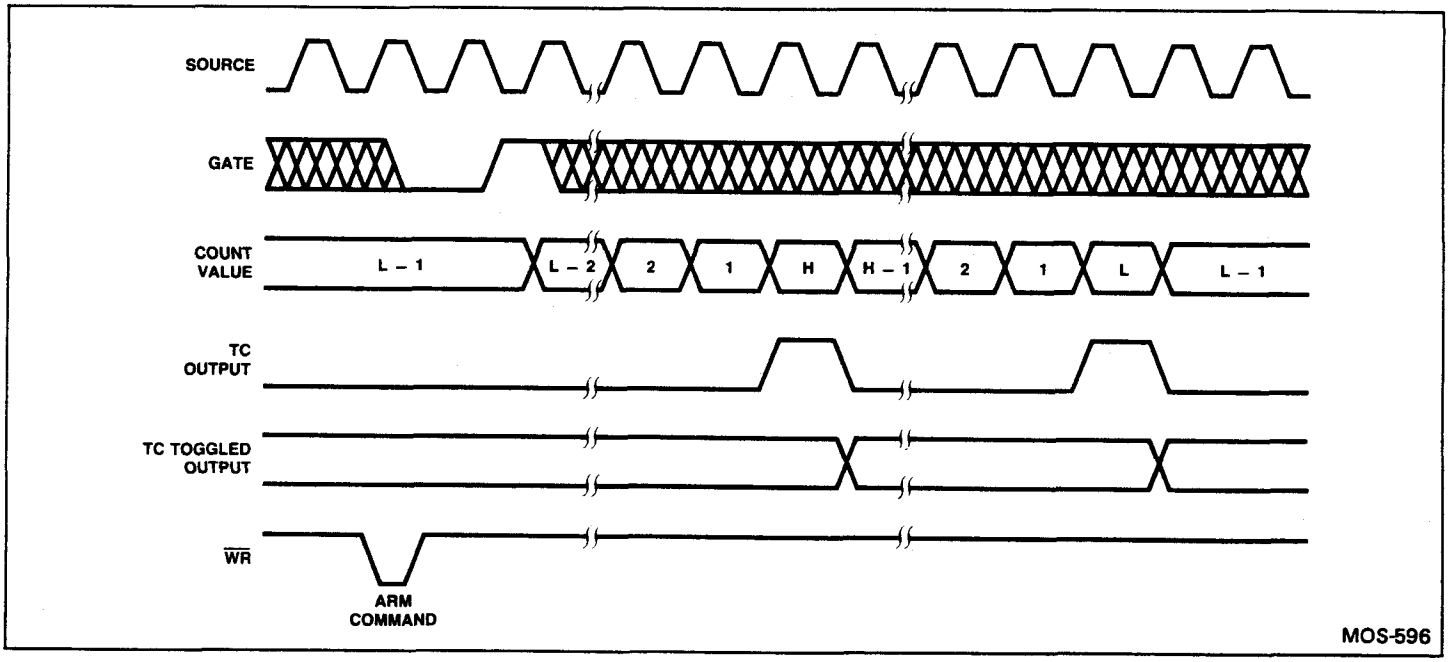



**Figure 1-17i. Mode I Waveforms**

**MODE J**
**Variable Duty Cycle Rate Generator with No Hardware Gating**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | X | X | X | X | X |

Mode J, shown in Figure 1-17j, will find the greatest usage in frequency generation applications with variable duty cycle requirements. Once armed, the counter will count continuously until it is issued a DISARM command. On the first TC, the counter will be reloaded from the Hold register. Counting will then proceed until the second TC at which time the counter will be reloaded from the Load register. Counting will continue, with the reload source alternating on each TC, until a DISARM command is issued to the counter. (The third TC reloads from the Hold register, the fourth TC reloads from the Load register, etc.) A variable duty cycle output can be generated by specifying the TC Toggled output in the Counter Mode register. The Load and Hold values then directly control the output duty cycle, with high resolution available when relatively high count values are used.

**MODE K**
**Variable Duty Cycle Rate Generator with Level Gating**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| LEVEL | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | X | X | X | X | X |

Mode K, shown in Figure 1-17k, is identical to Mode J except that source edges are only counted when the Gate is active. The counter must be armed for counting to occur. Once armed, the counter will count all source edges which occur while the Gate is active and disregard those source edges which occur while the Gate is inactive. This permits the Gate to turn the count process on and off. As with Mode J, the reload source used will alternate on each TC, starting with the Hold register on the first TC after any ARM command. When the TC Toggled output is used, this mode allows the Gate to modulate the duty cycle of the output waveform. It can affect both the high and low portions of the output waveform.

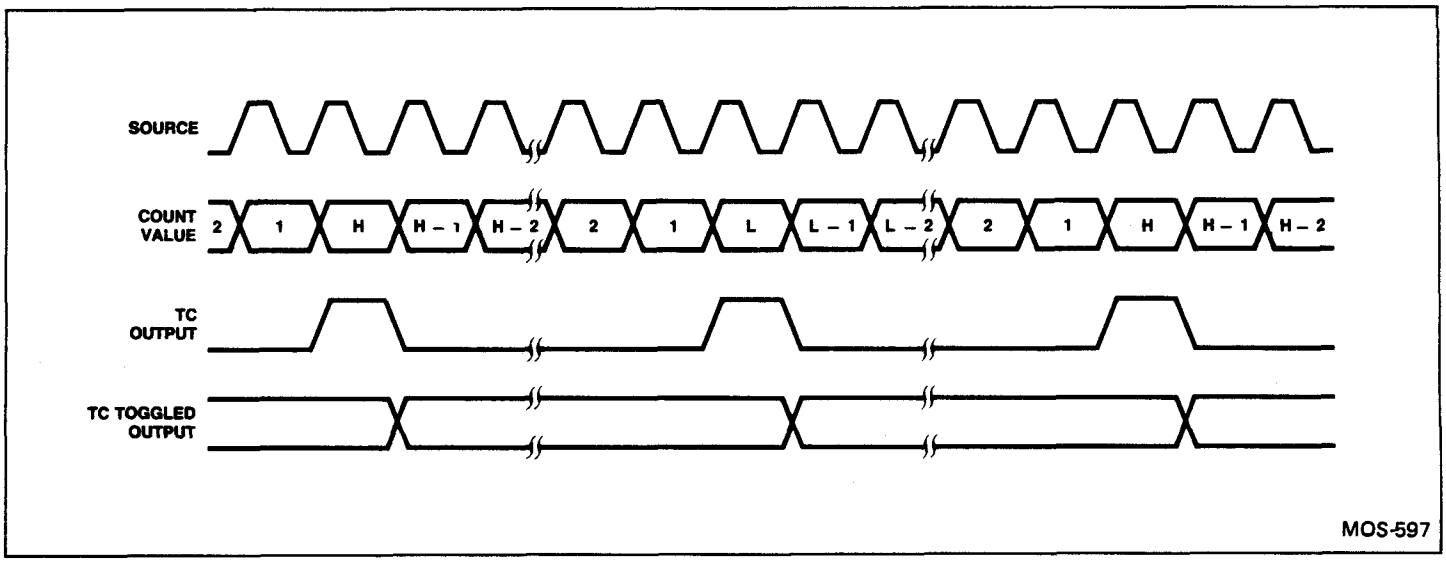



**Figure 1-17j. Mode J Waveforms**

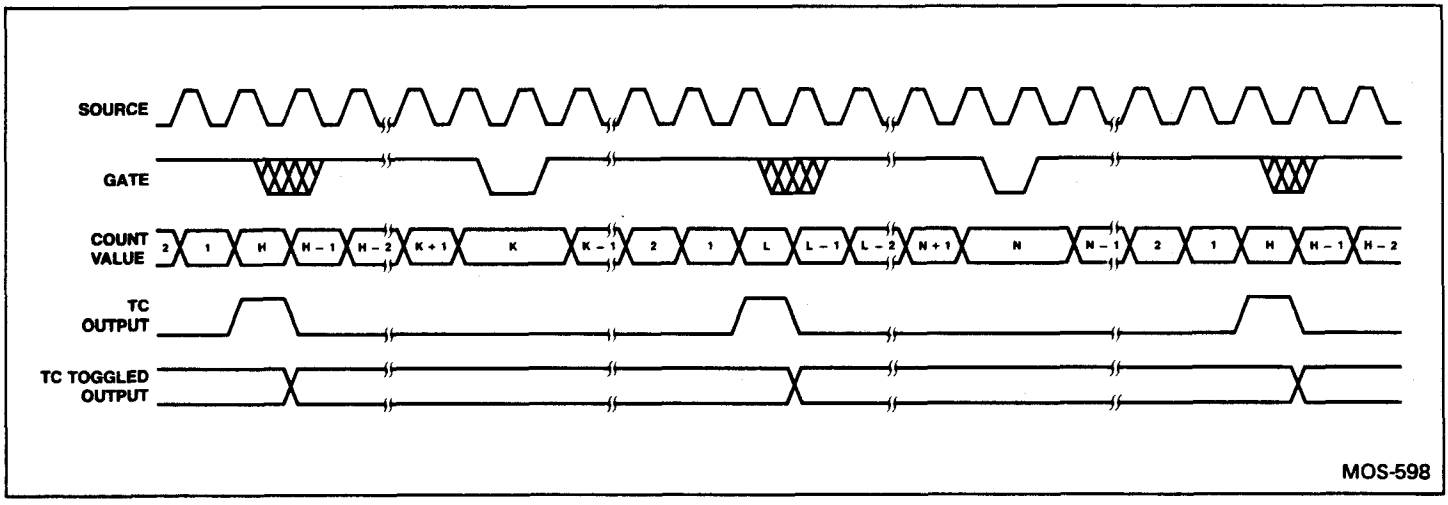



**Figure 1-17k. Mode K Waveforms**

**MODE L**
**Hardware-Triggered Delayed Pulse One-Shot**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| EDGE | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | X | X | X | X | X |

Mode L, shown in Figure 1-17l, is similar to Mode J except that counting will not begin until a Gate edge is applied to an armed counter. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. The counter will start counting source edges after the triggering Gate edge and counting will proceed until the second TC. Note that after application of a triggering Gate edge, the Gate input will be disregarded for the remainder of the count cycle. This differs from Mode K, where the gate can be modulated throughout the count cycle to stop and start the counter. On the first TC after application of the triggering Gate edge, the counter will be reloaded from the Hold register. On the second TC, the counter will be reloaded from the Load register and counting will stop until a new gate edge is issued to the counter. Note that unlike Mode K, new Gate edges are required after every second TC to continue counting.

**MODE N**
**Software-Triggered Strobe with Level Gating and Hardware Retriggering**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| LEVEL | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | X | X | X | X | X |

Mode N, shown in Figure 1-17n, provides a software-triggered strobe with level gating that is also hardware retriggerable. The counter must first be issued an ARM command before counting can occur. Once armed, the counter will count all source edges which occur while the gate is active and disregard those source edges which occur while the Gate is inactive. This permits the Gate to turn the count process on and off. After the issuance of an ARM command and the application of an active Gate, the counter will count to TC. Upon reaching TC, the counter will reload from the Load register and automatically disarm itself, inhibiting further counting. Counting will resume upon the issuance of a new ARM command. All active-going Gate edges issued to an armed counter will cause a retrigger operation. Upon application of the Gate edge, the counter contents will be saved in the Hold register. On the first qualified source edge after application of the retriggering gate edge the contents of the Load register will be transferred into the counter. Counting will resume on the second qualified source edge after the retriggering Gate edge. Qualified source edges are active-going edges which occur while the Gate is active.

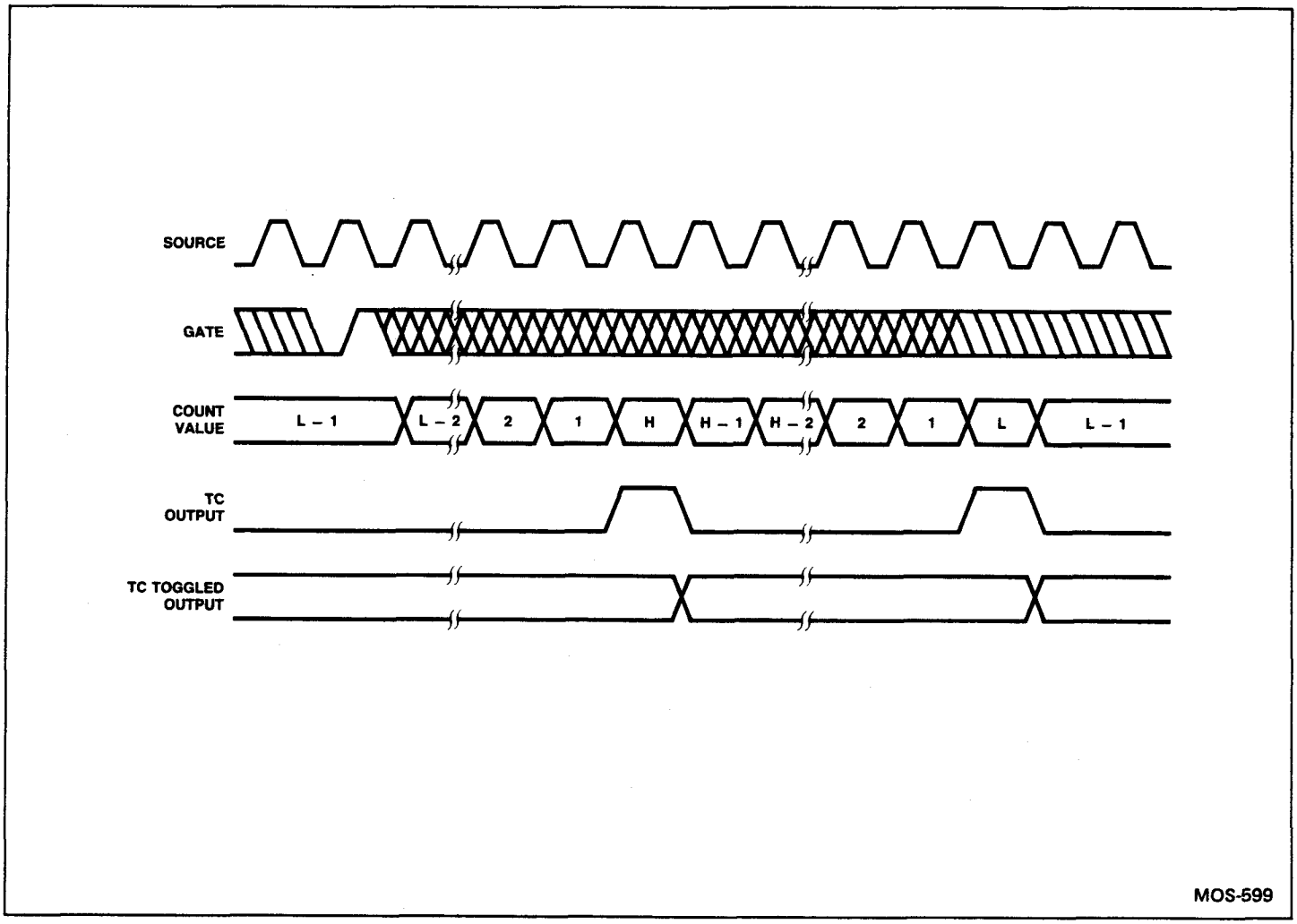



**Figure 1-17l. Mode L Waveforms**

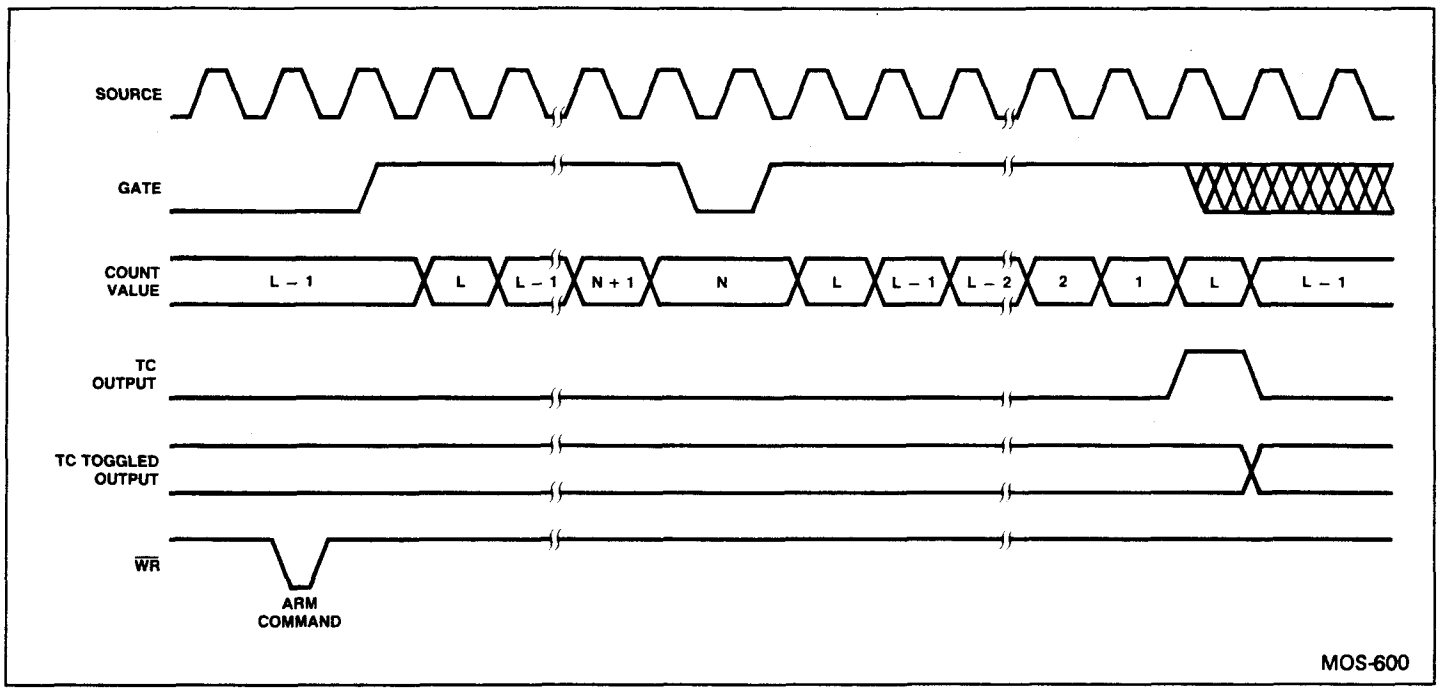



**Figure 1-17n. Mode N Waveforms**

## MODE O
### Software-Triggered Strobe with Edge Gating and Hardware Retriggering

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| EDGE | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | X | X | X | X | X |

Mode O, shown in Figure 1-17o, is similar to Mode N, except that counting will not begin until an active-going Gate edge is applied to an armed counter and the Gate level is not used to modulate counting. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. Irrespective of the Gate level, the counter will count all source edges after the triggering Gate edge until the first TC. On the first TC the counter will be reloaded from the Load register and disarmed. A new ARM command and a new Gate edge must be applied in that order to initiate a new counting cycle. Unlike Modes C, F, I and L, which disregard the Gate input once counting starts, in Mode O the count process will be retriggered on all active-going Gate edges, including the first Gate edge used to start the counter. On each retriggering Gate edge, the counter contents will be transferred into the Hold register. On the first source edge after the retriggering Gate edge the Load register contents will be transferred into the counter. Counting will resume on the second-source edge after a retrigger.

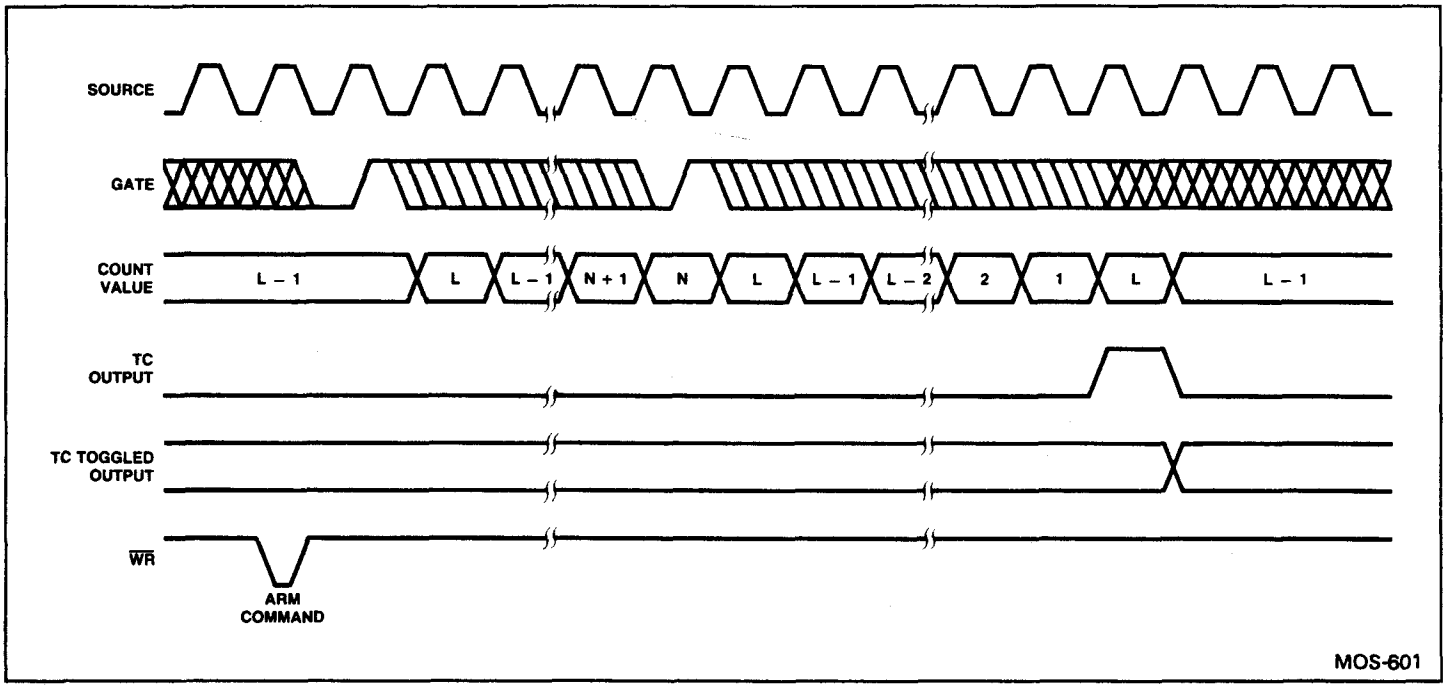



**Figure 1-17o. Mode O Waveforms**

**MODE Q**
**Rate Generator with Synchronization**
**(Event Counter with Auto-Read/Reset)**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| LEVEL | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | X | X | X | X | X |

Mode Q, shown in Figure 1-17q, provides a rate generator with synchronization or an event counter with auto-read/reset. The counter must first be issued an ARM command before counting can occur. Once armed, the counter will count all source edges which occur while the Gate is active and disregard those edges which occur while the Gate is inactive. This permits the Gate to turn the count process on and off. After the issuance of an ARM command and the application of an active Gate, the counter will count to TC repetitively. On each TC the counter will reload itself from the Load register. The counter may be retriggered at any time by presenting an active-going Gate edge to the Gate input. The retriggering Gate edge will transfer the contents of the counter into the Hold register. The first qualified source edge after the retriggering Gate edge will transfer the contents of the Load register into the Counter. Counting will resume on the second qualified source edge after the retriggering gate edge. Qualified source edges are active-going edges which occur while the Gate is active.

**MODE R**

**Retriggerable One-Shot**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| EDGE | | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | X | X | X | X | X |

Mode R, shown in Figure 1-17r, is similar to Mode Q, except that edge gating rather than level gating is used. In other words, rather than use the Gate level to qualify which source edges to count, Gate edges are used to start the counting operation. The counter must be armed before application of the triggering Gate edge; Gate edges applied to a disarmed counter are disregarded. After application of a Gate edge, an armed counter will count all source edges until TC, irrespective of the Gate level. On the first TC the counter will be reloaded from the Load register and stopped. Subsequent counting will not occur until a new Gate edge is applied. All Gate edges applied to the counter, including the first used to trigger counting, initiate a retrigger operation. Upon application of a Gate edge, the counter contents are saved in the Hold register. On the first source edge after the retriggering Gate edge, the Load register contents will be transferred into the counter. Counting will resume on the second source edge after the retriggering Gate edge.

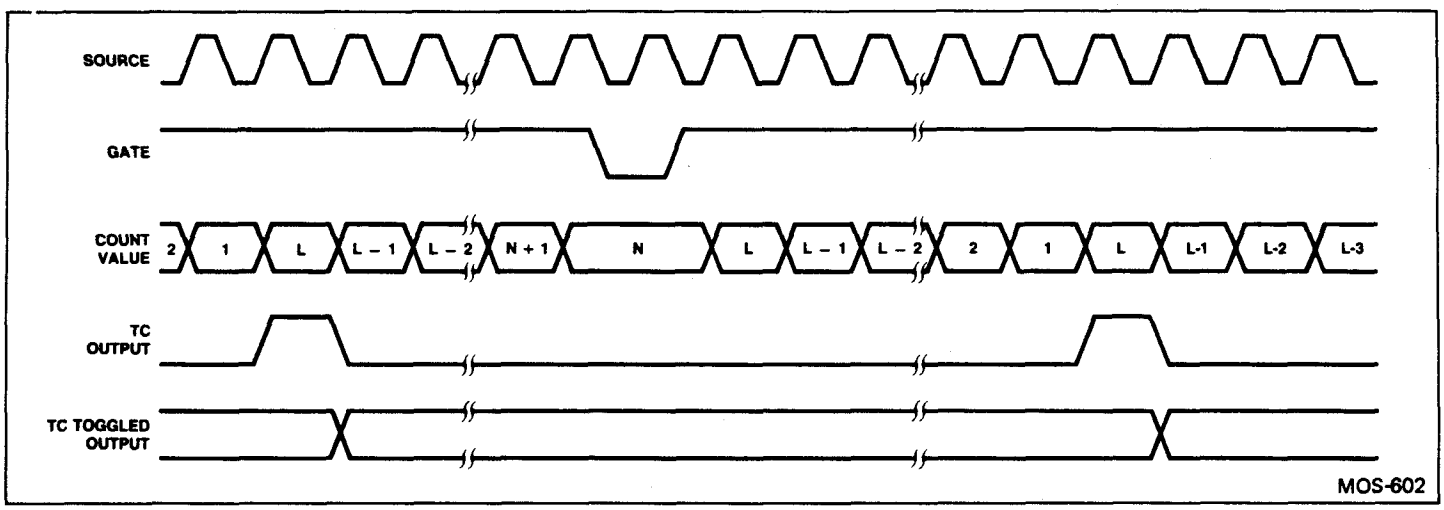



**Figure 1-17q. Mode Q Waveforms**

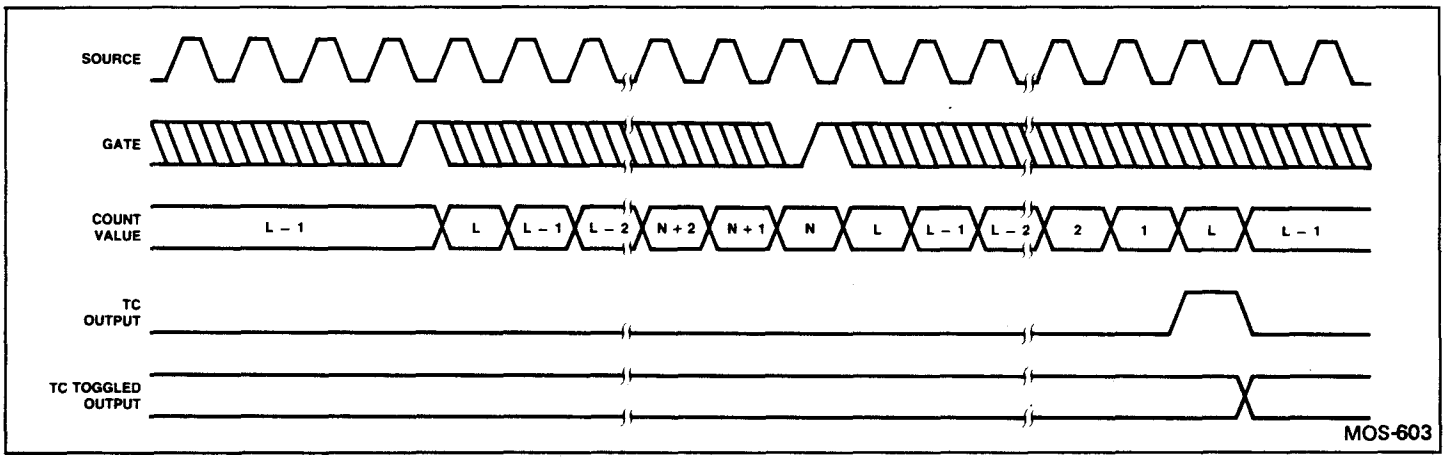



**Figure 1-17r. Mode R Waveforms**

## MODE S

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | X | X | X | X | X |

In this mode, the reload source for LOAD commands (irrespective of whether the counter is armed or disarmed) and for TC-initiated reloads is determined by the Gate input. The Gate input in Mode S is used only to select the reload source, not to start or modulate counting. When the Gate is Low, the Load register is used; when the Gate is High, the Hold register is used. Note the Low-Load, High-Hold mnemonic convention. Once armed, the counter will count to TC twice and then disarm itself. On each TC the counter will be reloaded from the reload source selected by the Gate. Following the second TC, an ARM command is required to start a new counting cycle. Mode S is shown in Figure 1-17s.

## MODE V

### Frequency-Shift Keying

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | X | X | X | X | X |

Mode V, shown in Figure 1-17v, provides frequency-shift keying modulation capability. Gate operation in this mode is identical to that in Mode S. If the Gate is Low, a LOAD command or a TC-induced reload will reload the counter from the Load register. If the Gate is High, LOADs and reloads will occur from the Hold register. The polarity of the Gate only selects the reload source; it does not start or modulate counting. Once armed, the counter will count repetitively to TC. On each TC the counter will reload itself from the register determined by the polarity of the Gate. Counting will continue in this manner until a DISARM command is issued to the counter. Frequency shift keying may be obtained by specifying a TC Toggled output mode in the Counter Mode register. The switching of frequencies is achieved by modulating the Gate.

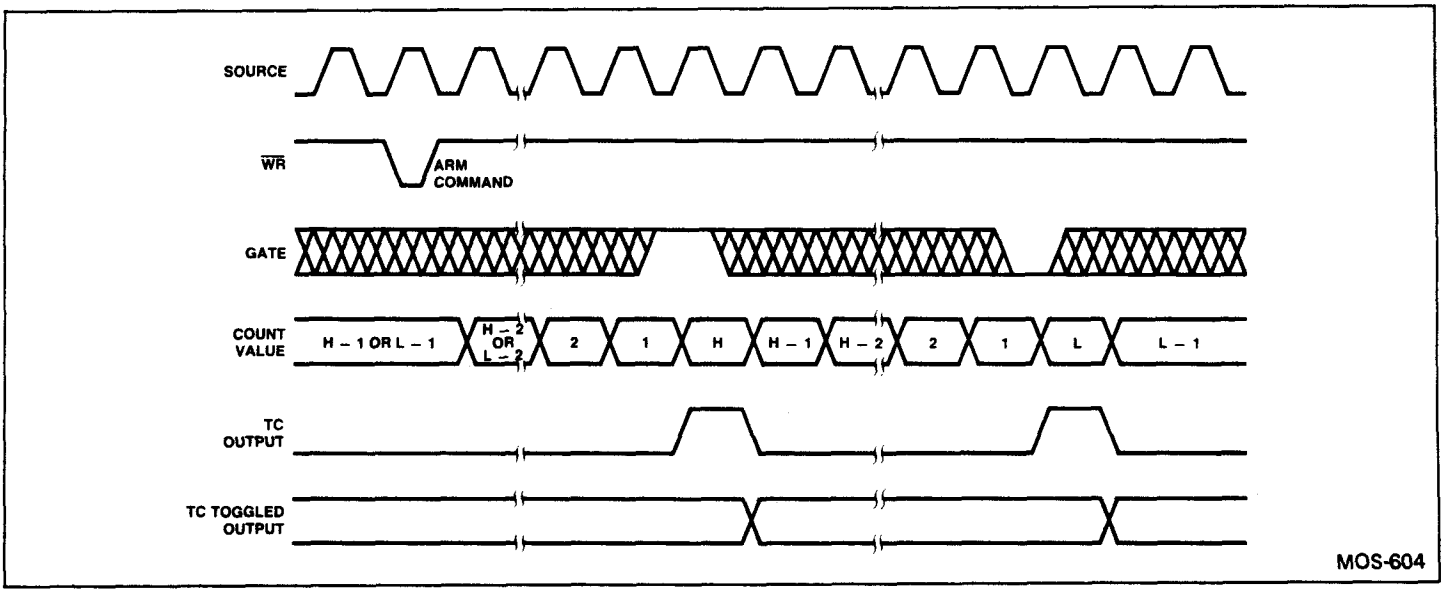



**Figure 1-17s. Mode S Waveforms**

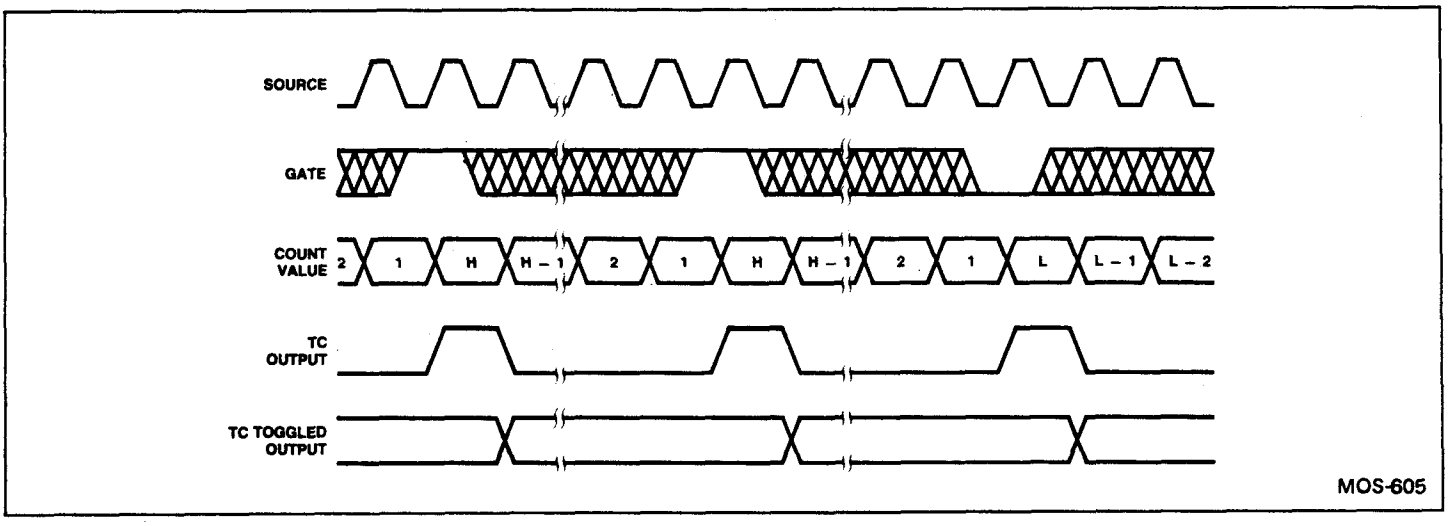



**Figure 1-17v. Mode V Waveforms**

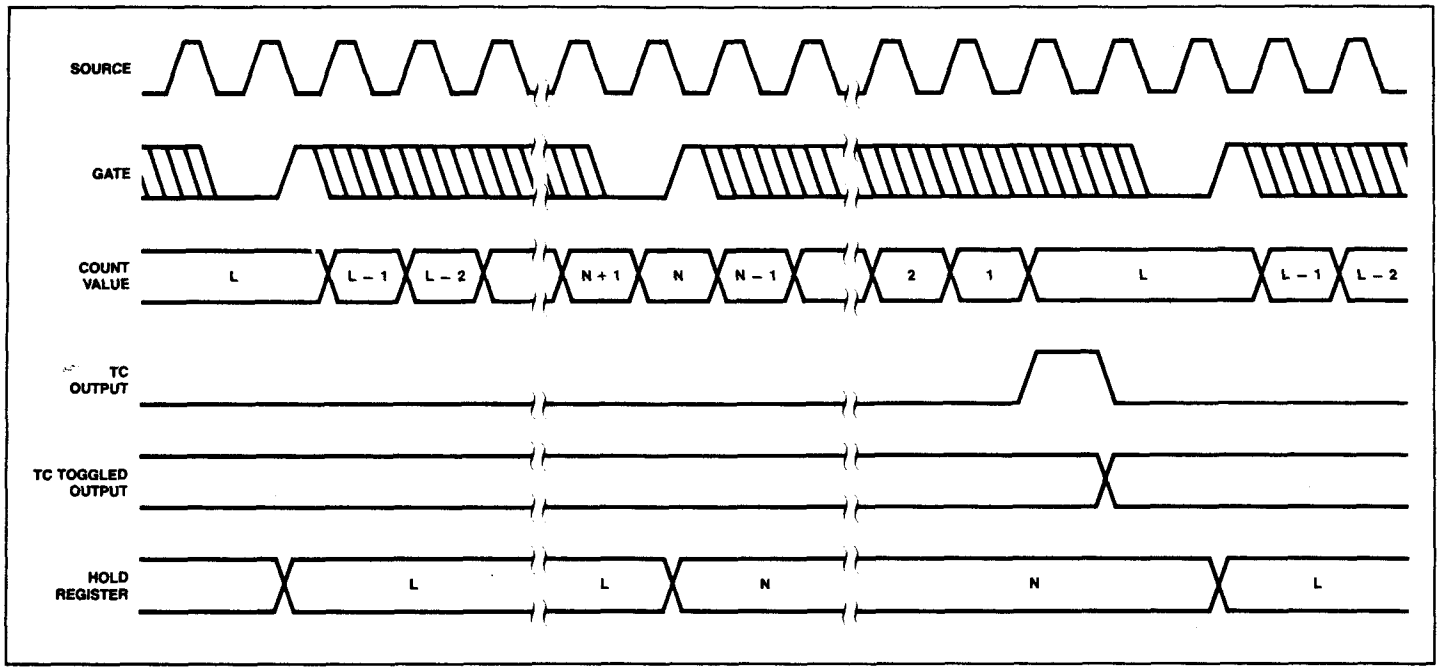



**Figure 1-17x. Mode X Waveforms**

## MODE X

**Hardware Save (available in Am9513A only)**

| CM15 | CM14 | CM13 | CM12 | CM11 | CM10 | CM9 | CM8 |
|---|---|---|---|---|---|---|---|
| | Edge | | X | X | X | X | X |

| CM7 | CM6 | CM5 | CM4 | CM3 | CM2 | CM1 | CM0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | X | X | X | X | X |

Mode X, shown in Figure 1-17x, provides a hardware sampling of the counter contents without interrupting the count. A Load and Arm command or a Load command followed by an Arm command is required to initialize the counter. Once armed, a Gate edge starts the counting operation; gate edges applied to a disarmed counter are disregarded. After application of the Triggering Gate edge the counter will count all qualified source edges until the first TC, irrespective of the gate level. All gate edges applied during the counting sequence will store the current count in the Hold register, but they will not interrupt the counting sequence. On each TC, the counter will be reloaded from the Load register and stopped. Subsequent counting requires a new triggering Gate edge; counting resumes on the first source edge following the triggering Gate edge.

Note: Mode X is only available in the Am9513'A' devices.

## COUNTER MODE CONTROL OPTIONS

Each Counter Logic Group includes a 16-bit Counter Mode (CM) register used to control all of the individual options available with its associated general counter. These options include output configuration, count control, count source and gating control. Figure 1-18 shows the bit assignments for the Counter Mode registers. This section describes the control options in detail. Note that generally each counter is independently configured and does not depend on information outside its Counter Logic Group. The Counter Mode register should be loaded only when the counter is Disarmed. Attempts to load the Counter Mode register when the counter is armed may result in erratic counter operation.

After power-on reset or a Master Reset command, the Counter Mode registers are initialized to a preset condition. The value entered is 0B00 hex and results in the following control configuration:

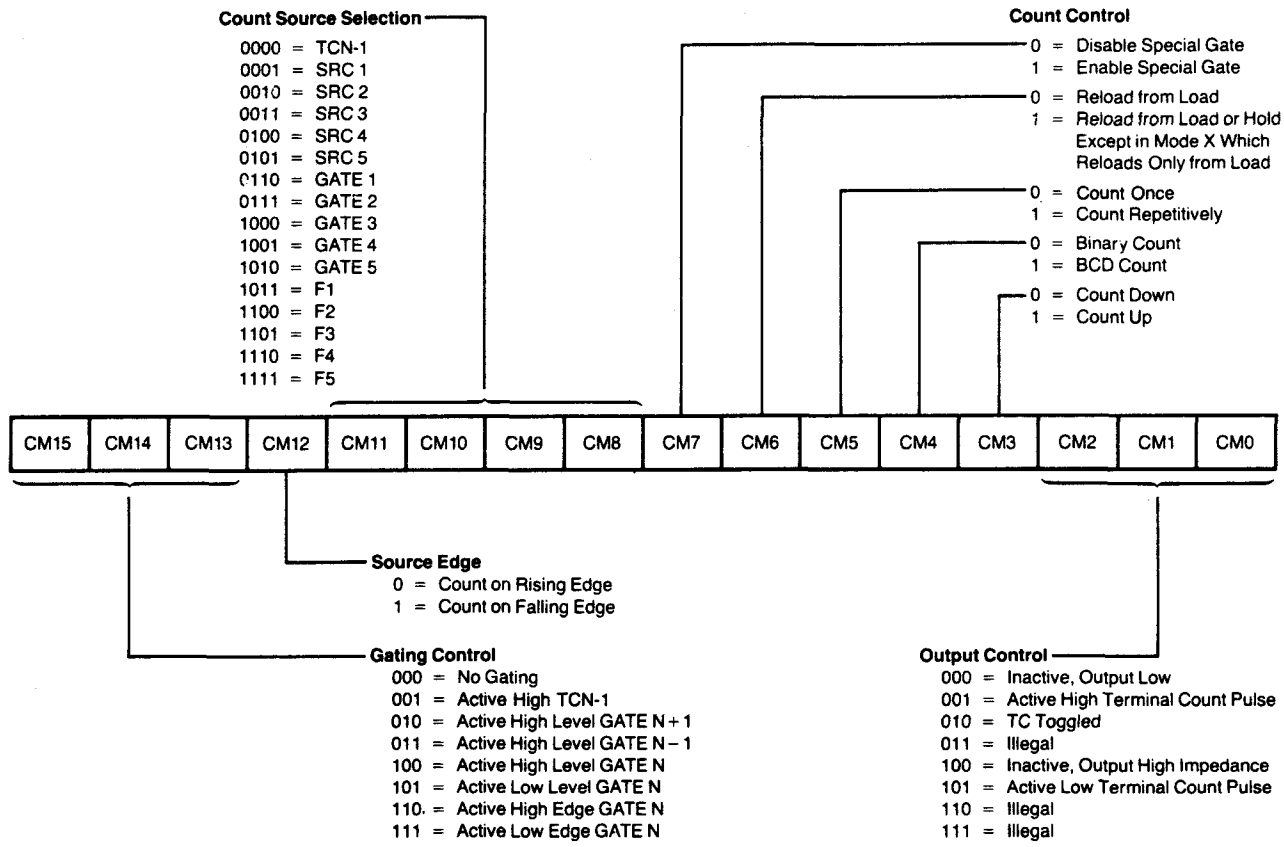
Output low impedance to ground
Count down
Count binary
Count once
Load register selected
No retriggering
F1 input source selected
Positive-true input polarity
No gating

### Output Control

Counter mode bits CM0 through CM2 specify the output control configuration. Figure 1-19 shows a schematic representation of the output control logic. The OUT pin may be off (a high impedance state), or it may be inactive with a low impedance to ground. The three remaining valid combinations represent the active High, active Low or TC Toggle output waveforms.

One output form available is called Terminal Count (TC) and represents the period in time that the counter reaches an equivalent value of zero. TC will occur on the next count when the counter is at 0001 for down counting, at 9999 (BCD) for BCD up counting or at FFFF (hex) for binary up counting. Figure 1-20 shows a Terminal Count pulse and an example context that generated it. The TC width is determined by the period of the counting source. Regardless of any gating input or whether the counter is Armed or Disarmed, the terminal count will go active for only one clock cycle. Figure 1-20 assumes active-high source polarity, counter armed, counter decrementing and an external reload value of K.

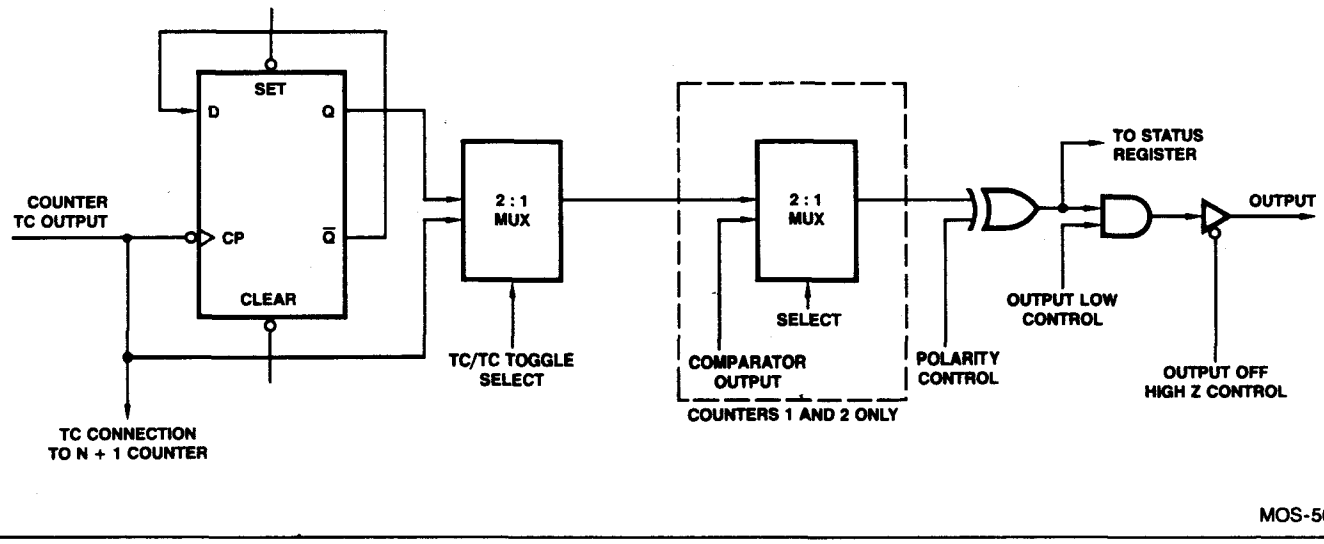
The counter will always be loaded from an external location when TC occurs; the user can choose the source location and the value. If a non-zero value is picked, the counter will never really attain a zero state and TC will indicate the counter state that would have been zero had no parallel transfer occurred.




Note: See Figure 1-17 for restrictions on Count Control and Gating Control bit combinations.

MOS-176

**Figure 1-18. Counter Mode Register Bit Assignments**




MOS-502A

**Figure 1-19. Output Control Logic**

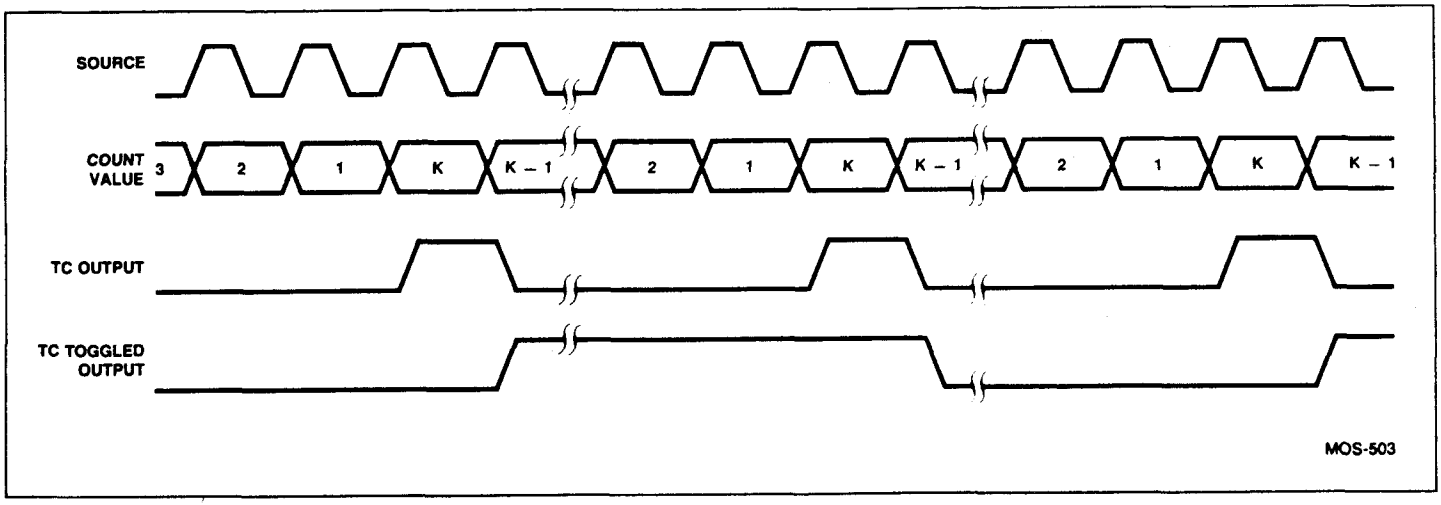



**Figure 1-20. Counter Output Waveforms**

The other output form, TC Toggled, uses the trailing edge of TC to toggle a flip-flop to generate an output level instead of a pulse. The toggle output is 1/2 the frequency of TC. The TC Toggled output will frequently be used to generate variable duty-cycle square waves in Operating Modes G through K.

In Mode L the TC Toggled output can be used to generate a one-shot function, with the delay to the start of the output pulse and the width of the output pulse separately programmable. With selection of the minimum delay to the start of the pulse, the output will toggle on the second source pulse following application of the triggering Gate edge.

Note that the TC Toggled output form contains no implication about whether the output is active-high or active-low. Unlike the TC output, which generates a transient pulse which can clearly be active-high or active-low, the TC Toggled output waveform only flips the state of the output on each TC. The sole criteria of whether the TC Toggled output is active-high or active-low is the level of the output at the start of the count cycle. This can be controlled by the Set and Clear Output commands. (See Figure 1-21.)

### TC (Terminal Count)

On each Terminal Count (TC), the counter will reload itself from the Load or Hold register. TC is defined as that period of time when the counter contents would have been zero had no reload occurred. Some special conditions apply to counter operation immediately before and during TC.

1. In the clock cycle before TC, an internal signal is generated that commits the counter to go to TC on the next count, and retriggering by a hardware Gate edge (Modes N, O, Q and R) or a software LOAD or LOAD-and-ARM command will not extend the time to TC. Note that the "next count" driving the counter to TC can be caused by the application of a count source edge (in level gating modes, the edge must occur while the gate is active, or it will be disregarded), by the application of a LOAD or LOAD-and-ARM command (see 2 below) or by the application of a STEP command.
2. If a LOAD or LOAD-and-ARM command is executed during the cycle preceding TC, the counter will immediately go to TC. If these commands are issued during TC, the TC state will immediately terminate.
3. When TC is active, the counter will always count the next source edge issued to it, even if it is disarmed or gated off during TC. This means that TC will never be active for longer than one count period and it may, in fact, be shorter if a STEP command or a LOAD or LOAD-and-ARM command is applied during TC (see item 2 above). This also means that a counter that is disarmed or stopped on TC is actually disarmed/stopped immediately following TC.

This may cause count sequences different from what a user might expect. Since the counter is always reloaded at the start of TC, and since it always counts at the end of TC, the counter contents following TC will differ by one from the reloaded value, irrespective of the operating mode used.

If the reloaded value was 0001 for down counting, 9999 (BCD) for BCD up counting or FFFF (hex) for binary up counting, the count at the end of TC will drive the counter into TC again regardless of whether the counter is gated off or disarmed. As long as these values are reloaded, the TC output will stay active. If a TC Toggled output is selected, it will toggle on each count. Execution of a LOAD, LOAD-and-ARM or STEP command with these counter contents will act the same as application of a source pulse, causing TC to remain active and a TC Toggled output to toggle.

### Count Control

Counter Mode bits CM3 through CM7 specify the various options available for direct control of the counting process. CM3 and CM4 operate independently of the others and control up/down and BCD/binary counting. They may be combined freely with other control bits to form many types of counting configurations. The other three bits and the Gating Control field interact in complex ways. Bit CM5 controls the repetition of the count process. When CM5 = 1, counting will proceed in the specified mode until the counter is disarmed. When CM5 = 0, the count process will proceed only until one full cycle of operation occurs. This may occur after one or two TC events. The counter is then disarmed automatically. The single or double TC requirement will depend on the state of other control bits. Note that even if the counter is automatically disarmed upon a TC, it always counts the count source edge which generates the trailing TC edge.

When TC occurs, the counter is always reloaded with a value from either the Load register or the Hold register. Bit CM6 specifies the source options for reloading the counter. When CM6 = 0, the contents of the Load register will be transferred into the counter at every occurrence of TC. When CM6 = 1, the counter reload location will be either the Load or Hold Register. The reload location in this case may be controlled externally by using a GATE pin (Modes S and V) or may alternate on each TC (Modes G through L). With alternating sources and with the TC Toggled output selected, the duty cycle of the output waveform is

controlled by the relative Load and Hold values and very fine resolution of duty cycle ratios may be achieved.

Bit CM7 controls the special gating functions that allow retriggering and the selection of Load or Hold sources for counter reloading. The use and definition of CM7 will depend on the status of the Gating Control field and bits CM5 and CM6.

### Hardware Retriggering

Whenever hardware retriggering is enabled (Modes N, O, Q and R) all active going Gate edges initiate retrigger operations. On application of the Gate edge, the counter contents will be transferred to the Hold register. On the first qualified source edge after application of the retriggering Gate edge, the Load register contents will be transferred into the counter. (Qualified source edges are edges which occur while the counter is gated on and Armed.)

This means that if level gating is used, the edge occurring on active-going gate transitions will initiate a retrigger. Similarly, when edge gating is enabled, an edge used to start the counter will also initiate a retrigger. The first count source edge applied after the Gate edge will not increment/decrement the counter but retrigger it.

If a Load, Load and Arm, or Step Command occur between the retriggering Gate edge and the first qualified source edge, it will be interpreted as a source edge and transfer the Load register contents into the counter. Thereafter, the counter will count all qualified source edges.

When some form of Gating is specified, CM7 controls hardware retriggering. In this case, when CM7 = 0 hardware retriggering does not occur; when CM7 = 1 the counter is retriggered any time an active-going Gate edge occurs. Retriggering causes the counter value to be saved in the Hold register and the Load register contents to be transferred into the counter.

When No Gating is specified, the definition of CM7 changes. In this case, when CM7 = 0 the Gate input has no effect on the counting; when CM7 = 1 the Gate input specifies the source (selecting either the Load or Hold register) used to reload the counter when TC occurs. Figure 1-16 shows the various available control combinations for these interrelated bits.

### Count Source Selection

Counter Mode bits CM8 through CM12 specify the source used as input to the counter and the active edge that is counted. Bit CM12 controls the polarity for all the sources; logic zero counts rising edges and logic one counts falling edges. Bits CM8 through CM11 select 1 of 16 counting sources to route to the counter input. Five of the available inputs are internal frequencies derived from the internal oscillator (see Figure 1-15 for frequency assignments). Ten of the available inputs are interface pins; five are labeled SRC and five are labeled GATE.

The 16th available input is the TC output from the adjacent lower-numbered counter. (The Counter 5 TC wraps around to the Counter 1 input.) This option allows internal concatenating that permits very long counts to be accumulated. Since all five counters may be concatenated, it is possible to configure a counter that is 80-bits long on one Am9513 chip. When TCN−1 is the source, the count ripples between the connected counters. External connections can also be made, and can use the toggle bit for even longer counts. This is easily accomplished by selecting a TC Toggled output mode and wiring OUTN to one of the SRC inputs.

### Gating Control

Counter Mode bits CM15, CM14, CM13 specify the hardware gating options. When "no gating" is selected (000) the counter will proceed unconditionally as long as it is armed. For any other gating mode, the count process is conditioned by the specified gating configuration.

For a code of 100 in this field, counting can proceed only when the pin labeled GATEN associated with Counter N is at a logic high level. When it goes low, counting is simply suspended until the Gate goes high again. A code of 101 performs the same function with an opposite active polarity. Codes 010 and 011 offer the same function as 100, but specify alternate input pins as Gating Sources. This allows any of three interface pins to be used as gates for a given counter. On Counter 4, for example, pin 34, pin 35 or pin 36 may be used to perform the gating function. This also allows a single Gate pin to simultaneously control up to three counters. Counters 1 and 5 are considered adjacent when using TCN − 1 (001), Gate N + 1 (010) and Gate N − 1 (011) controls.

For codes of 110 or 111 in this field, counting proceeds after the specified active Gate edge until one or two TC events occur. Within this interval the Gate input is ignored, except for the retriggering option. When repetition is selected, a cycle will be repeated as soon as another Gate edge occurs. With repetition selected, any Gate edge applied after TC goes active will start a new count cycle. Edge gating is useful when implementing a digital single-shot since the gate can serve as a convenient firing trigger.

A 001 code in this field selects the TC output from the adjacent lower-numbered counter as the gate. This is useful for synchronous counting when adjacent counters are concatenated.

## COMMAND DESCRIPTIONS

The command set for the Am9513 allows the host processor to customize and manage the operating modes and features for particular applications, to initialize and update both the internal data and control information, and to manipulate operating bits during operation. Commands are entered directly into the 8-bit Command register by writing into the Control port (see Figure 1-8).

All available commands are described in the following text. Figure 1-21 summarizes the command codes and includes a brief description of each function. Figure 1-22 shows all the unused code combinations; unused codes should not be entered into the Command register since undefined activities may occur.

Six of the command types are used for direct software control of the counting process and they each contain a 5-bit S field. In a linear-select fashion, each bit in the S field corresponds to one of the five general counters (S1 = Counter 1, S2 = Counter 2, etc.). When an S bit is a one, the specified operation is performed on the counter so designated; when an S bit is a zero, no operation occurs for the corresponding counter. This type of command format has three basic advantages. It saves host software by allowing any combination of counters to be acted on by a single command. It allows simultaneous action on multiple counters where synchronization of commands is important. It allows counter-specific service routines to control individual counters without needing to be aware of the operating context of other counters.

Three of the commands use a 3-bit binary code (N4, N2, N1) to identify the affected counter (a 001 programs counter 1, etc.). Unlike the previously mentioned commands, these commands allow you to program only one counter at a time.

| Command Code | | | | | | | | Command Description |
|---|---|---|---|---|---|---|---|---|
| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 | |
| 0 | 0 | 0 | E2 | E1 | G4 | G2 | G1 | Load Data Pointer register with contents of E and G fields. (G ≠ 000, G ≠ 110) |
| 0 | 0 | 1 | S5 | S4 | S3 | S2 | S1 | Arm counting for all selected counters |
| 0 | 1 | 0 | S5 | S4 | S3 | S2 | S1 | Load contents of specified source into all selected counters |
| 0 | 1 | 1 | S5 | S4 | S3 | S2 | S1 | Load and Arm all selected counters* |
| 1 | 0 | 0 | S5 | S4 | S3 | S2 | S1 | Disarm and Save all selected counters |
| 1 | 0 | 1 | S5 | S4 | S3 | S2 | S1 | Save all selected counters in Hold register |
| 1 | 1 | 0 | S5 | S4 | S3 | S2 | S1 | Disarm all selected counters |
| 1 | 1 | 1 | 0 | 1 | N4 | N2 | N1 | Set Toggle out (High) for counter N (001 ≤ N ≤ 101) |
| 1 | 1 | 1 | 0 | 0 | N4 | N2 | N1 | Clear Toggle out (Low) for counter N (001 ≤ N ≤ 101) |
| 1 | 1 | 1 | 1 | 0 | N4 | N2 | N1 | Step counter N (001 ≤ N ≤ 101) |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | Set MM14 (Disable Data Pointer Sequencing) |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | Set MM12 (Gate off FOUT) |
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | Set MM13 (Enter 16-bit bus mode) |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | Clear MM14 (Enable Data Pointer Sequencing) |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | Clear MM12 (Gate on FOUT) |
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | Clear MM13 (Enter 8-bit bus mode) |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | Enable Prefetch for Write operations (Am9513'A' only) |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | Disable Prefetch for Write operations (Am9513'A' only) |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | Master reset |

*Not to be used for asynchronous operations.

**Figure 1-21. Am9513 Command Summary**

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| 0 | 0 | 0 | X | X | 1 | 1 | 0 |
| 0 | 0 | 0 | X | X | 0 | 0 | 0 |
| * 1 | 1 | 1 | 1 | 1 | X | X | X |

*Unused except when XXX = 111, 001 or 000.

**Figure 1-22. Am9513 Unused Command Codes**

### Arm Counters

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | S5 | S4 | S3 | S2 | S1 |

Description: Any combination of counters, as specified by the S field, will be enabled for counting. A counter must be armed before counting can commence. Once armed, the counting process may be further enabled or disabled using the hardware gating facilities. This command can only arm or do nothing for a given counter; a zero in the S field does not disarm the counter.

ARM and DISARM commands can be used to gate counter operation on and off under software control. DISARM commands entered while a counter is in the TC state will not take effect until the counter leaves TC. This ensures that the counter never latches up in a TC state. (The counter may leave the TC state because of application of a count source edge; execution of a LOAD or LOAD-and-ARM command; or execution of a STEP command.)

In modes which alternate reload sources (Modes G-L), the ARMing operation is used as a reset for the logic which determines which reload source to use on the upcoming TC. Following each ARM or LOAD-and-ARM command, a counter in one of these modes will reload from the Hold register on the first TC and alternate reload sources thereafter (reload from the Load register on the second TC, the Hold register on the third, etc.).

### Load Counters

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | S5 | S4 | S3 | S2 | S1 |

Description: Any combination of counters, as specified in the S field, will be loaded with previously entered values. The source of information for each counter will be either the associated Load register or the associated Hold register, as determined by the operating configuration in the Mode register. The Load/Hold contents are not changed. This command will cause a transfer independent of any current operating configuration for the counter. It will often be used as a software retrigger, or as counter initialization prior to active hardware gating.

If a LOAD or LOAD-and-ARM command is executed during the cycle preceding TC, the counter will go immediately to TC. This occurs because the LOAD operation is performed by generating a pseudo-count pulse, internal to the Am9513, and the Am9513 is expecting to go into TC on the next count pulse. The reload source used to reload the counter will be the same as that which would have been used if the TC were generated by a source edge rather than by the LOAD operation.

Execution of a LOAD or LOAD-and-ARM command while a counter is in TC will cause the TC to end. For Armed counters in

all modes except S or V, the LOAD source used will be that to be used for the upcoming TC. (The LOADing operation will not alter the selection of reload source for the upcoming TC.) For Disarmed counters in modes except S or V, the reload sources used will be the LOAD register. For modes S or V, the reload source will be selected by the GATE input, regardless of whether the counter is Armed or Disarmed.

Special considerations apply when modes with alternating reload sources are used (Modes G-L). If a LOAD command drives the counter to TC in these modes, the reload source for the next TC will be from the opposite reload location. In other words, the LOAD-generated TC will cause the reload sources to alternate just as a TC generated by a source edge would. Note that if a second LOAD command is issued during the LOAD-generated TC (or during any other TC, for that matter) the second LOAD command will terminate the TC and cause a reload from the source designated for use with the next TC. The second LOAD will not alter the reload source for the next TC since the second LOAD does not generate a TC; reload sources alternate on TCs only, not on LOAD commands.

**Load and Arm Counters***

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | S5 | S4 | S3 | S2 | S1 |

Description: Any combination of counters, as specified in the S field, will be first loaded and then armed. This command is equivalent to issuing a LOAD command and then an ARM command.

A LOAD-and-ARM command which drives a counter to TC generates the same sequence of operations as execution of a LOAD command and then an ARM command. In modes which disarm on TC (Modes A-C and N-O, and Modes G-I and S if the current TC is the second in the cycle) the ARM part of the LOAD-and-ARM command will re-enable counting for another cycle. In modes which alternate reload sources (Modes G-L) the ARMing operating will cause the next TC to reload from the HOLD register, irrespective of which reload source the current TC used.

*This command should not be used during asynchronous operations.

**Disarm Counters**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | S5 | S4 | S3 | S2 | S1 |

Description: Any combination of counters, as specified by the S field, will be disabled from counting. A disarmed counter will cease all counting independent of other control conditions. The only exception to this is that a counter in the TC state will always count once, in order to leave TC, before DISARMing. This count may be generated by a source edge, by a LOAD or LOAD-and-ARM command (the LOAD-and-ARM command will negate the DISARM command) or by a STEP command. A disarmed counter may be updated using the LOAD command and may be read using the SAVE command. A count process may be resumed using an ARM command. See the ARM command description for further details.

**Save Counters**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | S5 | S4 | S3 | S2 | S1 |

Description: Any combination of counters, as specified by the S field, will have their contents transferred into their associated Hold register. The transfer takes place without interfering with any counting that may be underway. This command will overwrite any previous Hold register contents. The SAVE command is designed to allow an accumulated count to be preserved so that it can be read by the host CPU at some later time.

**Disarm and Save Counters**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | S5 | S4 | S3 | S2 | S1 |

Description: Any combination of counters, as specified by the S field, will be disarmed and the contents of the counter will be transferred into the associated Hold registers. This command is identical to issuing a DISARM command followed by a SAVE command.

**Set TC Toggle Output**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | N4 | N2 | N1 |

$(001 \leq N \leq 101)$

Description: The initial output level for TC Toggle mode is set (High) for counter N selected by N4, N2, N1 = 001 (Counter 1) thru 101 (Counter 5) respectively. This command conditions the TC Toggle flip-flop (see Figure 1-19), but does not appear at the counter output unless TC Toggle mode (CM2, CM1, CM0 = 010) is selected.

**Clear TC Toggle Output**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | N4 | N2 | N1 |

$(001 \leq N \leq 101)$

Description: The initial output level for TC Toggle mode is Cleared (Low) for counter N selected by N4, N2, N1 = 001 (Counter 1) thru 101 (Counter 5) respectively. This command conditions the TC Toggle flip-flop (see Figure 1-19), but does not appear at the counter output unless TC Toggle mode (CM2, CM1, CM0 = 010) is selected.

**Step Counter**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | N4 | N2 | N1 |

$(001 \leq N \leq 101)$

Description: Counter N is incremented or decremented by one, depending on its operating configuration. If the Counter Mode register associated with the selected counter has its CM3 bit cleared to zero, this command will cause the counter to decrement by one. If CM3 is set to a logic high, this command will increment the counter by one. The STEP command will take effect even on a disarmed counter.

**Load Data Pointer Register**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | E2 | E1 | G4 | G2 | G1 |

$(G4, G2, G1 \neq 000, \neq 110)$

Description: Bits in the E and G fields will be transferred into the corresponding Element and Group fields of the Data Pointer register as shown in Figure 1-9. The Byte Pointer bit in the Data Pointer register is set. Transfers into the Data Pointer only

occur for G field values of 001, 010, 011, 100, 101 and 111. Values of 000 and 110 for G should not be used. See the "Setting the Data Pointer Register" section of this document for additional details.

**Disable Data Pointer Sequencing**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 |

Description: This command sets Master Mode bit 14 without affecting other bits in the Master Mode register. MM14 controls the automatic sequencing of the Data Pointer register. Disabling the sequencing allows repetitive host processor access to a given internal location without repetitive updating of the Data Pointer. MM14 may also be controlled by loading a full word into the Master Mode register.

**Enable Data Pointer Sequencing**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

Description: This command clears Master Mode bit 14 without affecting other bits in the Master Mode register. MM14 controls the automatic sequencing of the Data Pointer register. Enabling the sequencing allows sequential host processor access to several internal locations without repetitive updating of the Data Pointer. MM14 may also be controlled by loading a full word into the Master Mode register. See the "Data Pointer Register" section of this document for additional information on Data Pointer sequencing.

**Enable 16-Bit Data Bus**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 |

Description: This command sets Master Mode bit 13 without affecting other bits in the Master Mode register. MM13 controls the multiplexer in the data bus buffer. When MM13 is set, no multiplexing takes place and all 16 external data bus lines are used to transfer information into and out of the STC. MM13 may also be controlled by loading the full Master Mode register in parallel.

**Enable 8-Bit Data Bus**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 |

Description: This command clears Master Mode bit 13 without affecting other bits in the Master Mode register. MM13 controls the multiplexer in the data bus buffer. When MM13 is cleared, the multiplexer is enabled and 16-bit internal information is transferred eight bits at a time to the eight low-order external data bus lines. MM13 may also be controlled by loading the full Master Mode register in parallel.

**Gate Off FOUT**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 |

Description: This command sets Master Mode bit 12 without affecting other bits in the Master Mode register. MM12 controls the output state of the FOUT signal. When gated off, the FOUT line will exhibit a low impedance to ground. MM12 may also be controlled by loading the full Master Mode register in parallel.

**Gate On FOUT**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 |

Description: This command clears Master Mode bit 12 without affecting other bits in the Master Mode register. MM12 controls the output status of the FOUT signal. When MM12 is cleared, FOUT will become active and will drive out the selected and divided FOUT signal. MM12 may also be controlled by loading the full Master Mode register in parallel. When FOUT is gated on or off, a transient pulse may be generated on the FOUT signal.

**Disable Prefetch for Write Operations**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |

Description: This command disables the prefetch circuitry during Write operations (it does not affect Read operations). This reduces the write recovery time and allows the user to use block move instructions for initialization of the Am9513 registers. Once prefetch is disabled for writing, an Enable Prefetch for Write or a Reset command is necessary to re-enable the prefetch circuitry for writing. Note: This command is only available in Am9513'A' devices; it is an illegal command in the "non-A Am9513" device.

**Enable Prefetch for Write Operations**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |

Description: This command re-enables the prefetch circuitry for Write operations. It is used only to terminate the Disable Prefetch Command. Note: This command is only available in Am9513'A' devices; it is an illegal command in the "non-A Am9513" device.

**Master Reset**

Coding:

| C7 | C6 | C5 | C4 | C3 | C2 | C1 | C0 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

Description: The Master Reset command duplicates the action of the power-on reset circuitry. It disarms all counters, enters 0000 in the Master Mode, Load and Hold registers and enters 0B00 (hex) in the Counter Mode registers.

Following either a power-up or software reset, the LOAD command should be applied to all the counters to clear any that may be in a TC state. The Data Pointer register should also be set to a legal value, since reset does not initialize it. A complete reset operation is given in the following.

1. Using the procedure given in the "Command Initiation" section of this document, enter the FF (hex) command to perform a software reset.
2. Using the "Command Initiation" procedure, enter the LOAD command for all counters, opcode 5F (hex).
3. Using the procedure given in the "Setting the Data Pointer Register" section of this document, set the Data Pointer to a valid code. The legal Data Pointer codes are given in Figure 1-10.

The Master Mode, Counter Mode, Load and Hold registers can now be initialized to the desired values.

# Chapter 2
# Am9513A/Am9513 Interfacing

## Am9513 – CPU INTERFACING

The Am9513 is designed to interface easily to both the Am8080A/8085A 8-bit family of CPUs and to the AmZ8000 16-bit family of CPUs. Master Mode register bit MM13 allows the user to program the Am9513 data bus for either an 8- or 16-bit width, allowing the Am9513's data bus to be tailored to match that of the host CPU.

Figure 2-1 shows an interface between the Am9513 and an Am8085A CPU. The Am9513 is configured to appear in the CPU's I/O space; connecting the IO/$\overline{M}$ output of the CPU to the $\overline{G2A}$ input of the decoder and tying G1 high will memory-map the Am9513. In the configuration shown, the Am9513 operates with an 8-bit data bus. Master Mode register bit MM13 should be 0 and data bus pins DB13-DB15 should be tied high as shown in the diagram.

Figure 2-2 shows a suggested connection diagram between the Am9513 and an AmZ8001* or AmZ8002* CPU. In this diagram the Am9513 appears in both Regular and Special I/O space, by virtue of the decoding of status lines ST1-ST3. Status line ST0 should be decoded also if it is necessary to separate the Regular and Special I/O spaces. The AmZ8136 is a latched decoder which stores the address information on the rising edge of $\overline{AS}$, providing the Am9513 with a stable $\overline{CS}$ for the duration of the transfer. The Am25LS158 multiplexer generates $\overline{RD}$ and $\overline{WR}$ from the CPU's $\overline{DS}$ and R/$\overline{W}$ lines. For maximum data bandwidth between the CPU and the Am9513, Master Mode register bit MM13 should be set to 1 to configure the Am9513 for a 16-bit data bus width. This can be accomplished by writing command opcode FFEF (hex) to the Am9513 following each reset and power-up.

## CLOCK GENERATION

An internal oscillator is provided on the Am9513 for generation of timing frequencies to drive the source inputs for the five counters and the source for the FOUT pin. Note that a clock signal is not required for reads and writes to the Am9513. In applications which do not use the internal oscillator, the X2 input should be tied either High or Low to prevent accumulation of static charge. The X1 output is driven by an inverter contained in the Am9513 and accordingly, X1 should be left floating to avoid damaging the inverter's output stage.

Applications using the internal oscillator can drive the X1 and X2 inputs with an RC network, an external non-TTL level squarewave or a crystal. Figure 2-3 shows the recommended methods of connecting different frequency sources to the internal oscillator's input.

A crystal provides a highly accurate frequency source at moderate cost, and will usually be the preferred method of operation. The Am9513 is designed to use a crystal in parallel-resonant fundamental mode operation using the connection diagram shown in Figure 2-3a. Most series-resonant crystals can also be used, but the oscillator frequency will be different by up to a few percent from the series resonant crystal's rated frequency. Two ceramic capacitors should be connected between X1 and X2 to ground to ensure proper crystal loading. (The crystal loading is the capacitance the crystal should be driving to ensure on-frequency operation and reliable oscillator startup). Although the crystal sees the capacitors on X1 and X2 in series, and neglects the ground connection in the center, the use of two capacitors stabilizes the bias on the crystal by referencing it to ground and provides superior performance over the one capacitor equivalent circuit. Ceramic capacitors are the best type for this application because of their stability over time and temperature and their superior high frequency characteristics.

An RC network provides a very low cost frequency source but may exhibit large frequency variations over recommended power supply and temperature ranges, negating much of the precision available in the Am9513's counters. The RC connection is shown in Figure 2-3b. Note that although there is an internal resistor between X1 and X2, because this internal resistance is quite high, an external resistor should always be used in the RC operating configurations.

*Z8001 and Z8002 are trademarks of Zilog, Inc.

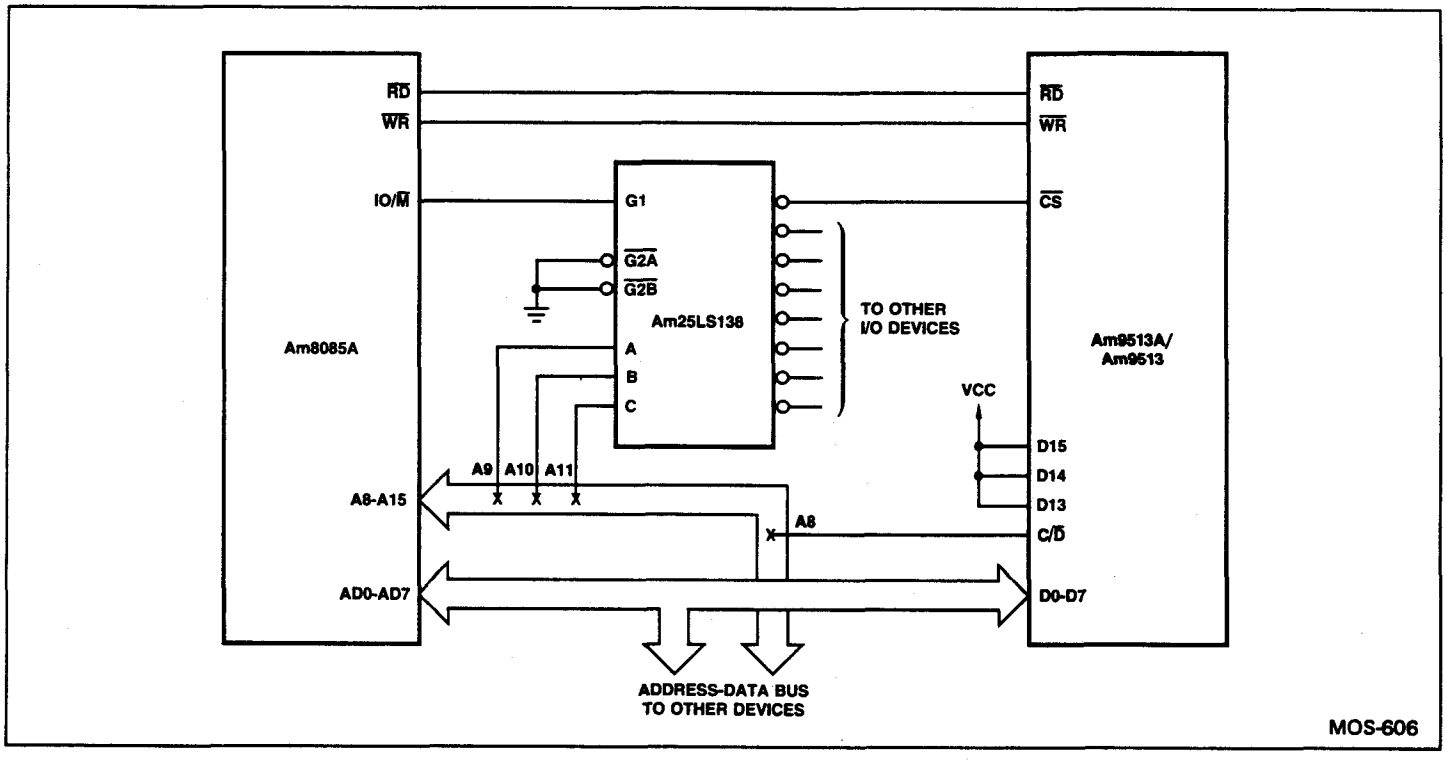



**Figure 2-1. Am9513 – Am8085 Interfacing**

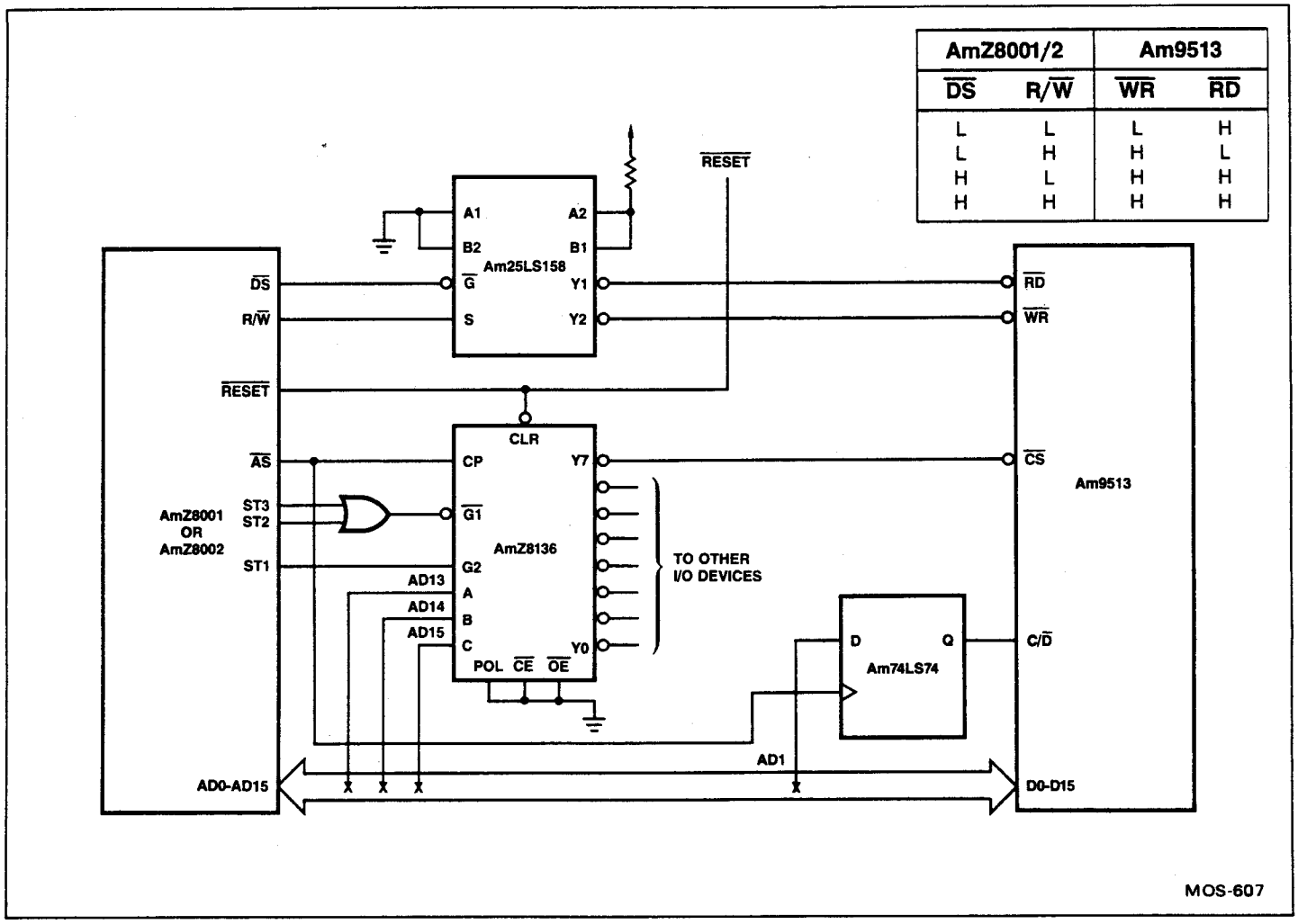
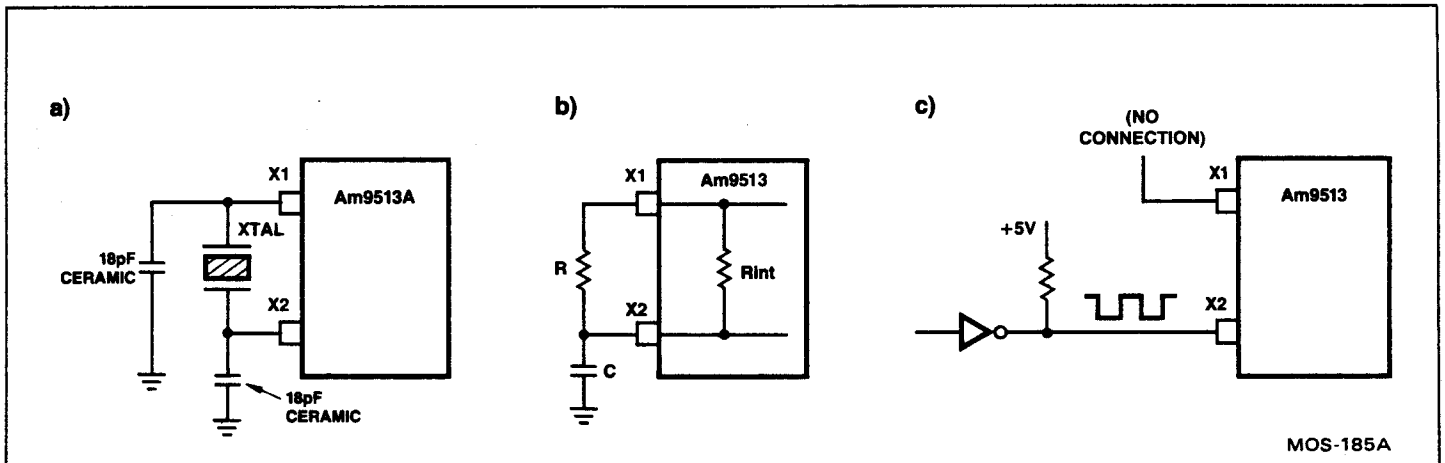



**Figure 2-2. AmZ8001/8002 – Am9513 Interface**




*Note: The Am9513A oscillator was changed from the Am9513. The capacitor values in previous designs should be changed to the values shown.

**Figure 2-3. Driving the X1 and X2 Inputs**

The Am9513 internal oscillator can also be driven by an external signal as shown in Figure 2-3c. The Am9513 Electrical Specification should be consulted for the voltage levels required on the X2 input to guarantee proper oscillator operation in this configuration. Most circuits can generate this non-TTL level using a pull-up resistor and a 74LS04 inverter, or equivalent. In some cases a pull-up resistor can be used to increase the high level output voltage of an MOS device, such as on the Am8085A CLK output, without the need for a bipolar buffer. Care must be taken in this bufferless circuit to choose a pull-up resistor low enough to meet the Am9513's high level voltage needs without choosing a resistor value so low that the Am8085A has to sink excessively large currents when pulling the CLK signal low.

## REGISTER ACCESS

### Information Transfer Protocols

The control signal configurations for all information transfers on the Am9513 data bus are summarized in Figure 2-4. The interface control logic assumes these conventions:

1. $\overline{RD}$ and $\overline{WR}$ are never active at the same time.
2. $\overline{RD}$, $\overline{WR}$ and C/$\overline{D}$ are ignored unless $\overline{CS}$ is Low.

The following discussion provides software oriented examples of Am9513 register accesses. Software examples are given for an Am8085 CPU with an 8-bit Am9513 data bus interface and for an AmZ8002 CPU with a 16-bit Am9513 data bus interface. The descriptions assume that the Am9513 Control port (CMDPRT) is located at address 12 (hex) and the Am9513 Data port (DATAPRT) is located at address 10 (hex). Later sections of this document present complete software listings for representative Am9513 applications.

### Software Initialization

Figure 2-5 shows a Z8000 Software Initialization Sequence for the 9513. It is important to note the "DUMMY" LOAD COUNTER COMMAND; this insures proper operation of the part. The 16-bit mode command is not used for 8-bit CPUs. The sequence then is to Reset the device; Load all counters; Command 16-bit mode; set Data Pointer to the Master Mode Register; set Master Mode Register to desired value; set Data Pointer to counter #1 Mode Register and initialize counters to desired mode of operation. Note $\overline{CS}$ must be high during power-up or the internal reset circuitry will not function correctly. This will result in part ignoring all commands issued to it except software reset.

### Command Initiation

Commands are issued to the Am9513 by writing the appropriate command code to the Am9513 Control Port. Figure 2-6 shows an example of command initiation, in this case opcode

| Signal Configuration | | | | Data Bus Operation |
|---|---|---|---|---|
| $\overline{CS}$ | C/$\overline{D}$ | $\overline{RD}$ | $\overline{WR}$ | |
| 0 | 0 | 0 | 1 | Transfer contents of register addressed by Data Pointer to the data bus. |
| 0 | 0 | 1 | 0 | Transfer contents of data bus to data register addressed by Data Pointer. |
| 0 | 1 | 0 | 1 | Transfer contents of Status register to data bus. |
| 0 | 1 | 1 | 0 | Transfer contents of data bus into Command register. |
| X | X | 1 | 1 | No transfer. |
| 1 | X | X | X | No transfer. |
| X | X | 0 | 0 | Illegal Condition. |

**Figure 2-4. Data Bus Transfers**

AD (hex), which saves the contents of Counters 1, 3 and 4 in their associated Hold registers. In both the Am8080A/8085A and AmZ8002 coding examples, the command is loaded into an internal CPU register and output to the appropriate port. Note that in the AmZ8002 case since a 16-bit data bus interface is assumed the upper byte of data output to the Command port must be FF (hex).

The procedure for executing a command is as follows:

1. Establish the appropriate command on the DB0-DB7 lines. Figure 1-21 lists the command codes. When using the Am9513 in 16-bit mode, data bus lines DB8-DB15 should be set high during the write operation. In 8-bit data bus mode, DB13-DB15 should be set high during the write operation.

```
MACRO8000:          Version 2.0    9/19/80
MACZ  9513INIT S,P,L,O,W,D
INIT


0000                              %THIS IS A SAMPLE INITIALIZATION SEQUENCE
0000                              %FOR THE AM9513 COUNTER TIMER
0000
0000                        MODULE 'INIT';
0000
0000                        CONST   CMDPRT=12H,
0000                                DATAPRT=10H;
0000
0000   2101 FFFF            INIT:   LD      R1,#FFFF;
0004   3B16 0012                    OUT     CMDPRT,R1;      %SEND RESET
0008   2101 FF5F                    LD      R1,#FF5F;
000C   3B16 0012                    OUT     CMDPRT,R1;      %LOAD ALL COUNTERS
0010   2101 FFEF                    LD      R1,#FFEF;
0014   3B16 0012                    OUT     CMDPRT,R1;      %COMMAND 16 BIT MODE
0018   2101 FF17                    LD      R1,#FF17;
001C   3B16 0012                    OUT     CMDPRT,R1;      %POINT TO MASTER MODE REG
0020   2101 2CEF                    LD      R1,#2CEF;
0024   3B16 0010                    OUT     DATAPRT,R1;     %MASTER MODE SETTING
0028   2101 FF01                    LD      R1,#FF01;
002C   3B16 0012                    OUT     CMDPRT,R1;      %POINT TO CNTR 1 MODE REG
0030
0030                              END.
```

**Figure 2-5. Am9513 Initialization**

```
0100                    ORG     100H
                        ;
                        ;       AM9513 PORT ADDRESSES
                        ;
0012 =          CMDPRT  EQU     012H
0010 =          DATAPRT EQU     010H
                        ;
                        ;       AM9513 COMMAND INITIATION
                        ;
0100 3EAD               MVI     A,0ADH  ;SAVE CTRS. 1 3 & 4
0102 D312               OUT     CMDPRT
                        ;
                        ;
                        PAGE
```

**a) 8080 Code**

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 1


0000                            PROGRAM AM9513_EXAMPLES;
0000                            ORIGIN  0H;
0000                            %
0000                            %
0000                            %       AM9513 PORT ADDRESSES
0000                            %
0000                    CONST   CMDPRT=12H,
0000                            DATAPRT=10H;
0000                            %
0000                    AM9513_EXAMPLES:
0000                            %
0000                            %       AM9513 COMMAND INITIATION
0000                            %
0000    2102 FFAD               LD      R2,0FFADH;      %SAVE CTRS. 1 3 & 4
0004    3B26 0012               OUT     CMDPRT,R2;
0008                            %
0008                            %
0008                            EJECT;
```

**b) AmZ8000 Code**

**Figure 2-6. Command Initiation Software**

2. Establish a High on the C/$\overline{D}$ input.
3. Establish a Low on the $\overline{CS}$ input.
4. Establish a Low on the $\overline{WR}$ input.
5. Sometime after the minimum $\overline{WR}$ low pulse duration has been achieved, drive $\overline{WR}$ high, taking care the $\overline{CS}$, C/$\overline{D}$ and data setup times are met (see Timing Diagram).
6. After meeting the required $\overline{CS}$, C/$\overline{D}$ and data hold times, these signals can be changed (see Timing Diagram).

A new read or write operation to the Am9513 should not be performed until the write recovery time is met (see Timing Diagram in Electrical Specification.)

### Setting the Data Pointer Register

The Data Pointer register selects which internal Am9513 register is to be accessed through the Data port. Setting the Data Pointer register automatically sets the Byte Pointer to 1, indicating a least significant byte is expected for 8-bit data bus interfacing. If Master Mode register bit MM14 = 0, the Data Pointer will automatically sequence through one of the cycles shown in Figure 1-11 after reading or writing each register, allowing sequential access to internal registers. If MM14 = 1, auto-sequencing is disabled and a single internal register can be repetitively accessed without reloading the Data Pointer. For convenience, bit MM14 can be set or cleared by software command.

The Pointer is set as follows:

1. Using Figures 1-9 and 1-10, select the appropriate Data Pointer Group and Element codes for the register to be accessed. Note that two codes are provided for the Hold registers, to accommodate both the Hold Cycle and Element Cycle autosequencing modes shown in Figure 1-11. If autosequencing is disabled, either Hold code may be used.
2. Using the "Writing to the Command Register" procedure given above, write the appropriate "Load Data Pointer" command to the Command register.

```
                        ;
                INTSR:          ;INTERRUPT SERVICE ROUTINE
                        ;
                        ;       DISABLE INTERRUPTS
                        ;
0104 F3                 DI
                        ;
                        ;       SET DATA POINTER TO COUNTER 1 HOLD REG
                        ;
0105 3E19               MVI     A,019H
0107 D312               OUT     CMDPRT
                        ;
                        ;       ENABLE AUTO-SEQUENCING
                        ;
0109 3EE0               MVI     A,0E0H
010B D312               OUT     CMDPRT
                        ;
                        ;       CODE TO ACCESS REGISTERS
                        ;
                        ;       DISABLE AUTO-SEQUENCING
                        ;
010D 3EE8               MVI     A,0E8H
010F D312               OUT     CMDPRT
                        ;
                        ;       ENABLE INTERRUPTS AND RETURN
                        ;
0111 FB                 EI
0112 C9                 RET
                        ;
                        ;
                        PAGE
```

a) 8080 Code

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 2

0008                            %
0008                    INTSR:  %       INTERRUPT SERVICE ROUTINE
0008                            %
0008    7C00                    DI      NVI,VI;
000A                            %
000A                            %       SET DATA POINTER TO COUNTER 1 HOLD REG.
000A                            %
000A    2102 FF19               LD      R2,0FF19H;
000E    3B26 0012               OUT     CMDPRT,R2;
0012                            %
0012                            %       ENABLE AUTO-SEQUENCING
0012                            %
0012    2102 FFE0               LD      R2,0FFE0H;
0016    3B26 0012               OUT     CMDPRT,R2;
001A                            %
001A                            %       CODE TO ACCESS REGISTERS
001A                            %
001A                            %       DISABLE AUTO-SEQUENCING
001A                            %
001A    2102 FFE8               LD      R2,0FFE8H;
001E    3B26 0012               OUT     CMDPRT,R2;
0022                            %
0022                            %       ENABLE INTERRUPTS AND RETURN
0022                            %
0022    7C04                    EI      NVI,VI;
0024    7B00                    IRET;
0026                            %
0026                            %
0026                            EJECT;
```

b) AmZ8000 Code

**Figure 2-7. Am9513 Interrupt Service Routine**

In many systems the Am9513 counters will be serviced by interrupt routines. In such systems, it is important that the Am9513 service routines not be interrupted by another Am9513 service routine while register accesses are occurring. Consider, for example, an interrupt service routine which reads the Hold register value in the Counter 1 logic group. This routine will set the Data Pointer register and read the Hold register value. Consider the sequence of events which would occur if, after this routine set the Data Pointer registers, but before it read the Hold register, it was interrupted by a second Am9513 interrupt routine. This second routine might, for example, read the Counter 3 logic group Hold register value. When this second interrupt routine finishes, it returns control to the last half of the first interrupt routine. Because the second routine has changed the Data Pointer register, the first routine will not read the Hold register 1 contents. As can be seen from the above scenario, the sequence of operations of setting the Data Pointer register and accessing internal register locations must not be interrupted by another Am9513 service routine.

One way of ensuring that this restriction is met is to disable interrupts before setting the Data Pointer and not enabling interrupts until the register accesses are performed. Note that when auto-sequencing is used, interrupts should not be enabled until all registers have been accessed. An alternative method of meeting this restriction is to use software semaphores to prevent nesting of Am9513 service routines.

Figure 2-7 shows sample interrupt service routines which set the Data Pointer register to point to Counter 1's Hold register and enable Hold cycle auto-sequencing by clearing MM14. In the AmZ8002 case, a 16-bit data bus interface is assumed, requiring that the upper command byte be FF (hex). In the coding examples given interrupts are disabled and enabled by software command. Since the AmZ8002 architecture loads a new Flag and Control Word (FCW) when responding to an Interrupt request, the FCW loaded can disable further interrupts. This provides an alternative interrupt inhibiting mechanism for AmZ8002 systems and may be used in lieu of the software commands.

### Reading the Status Register

The Am9513 Status register can be read either through the Control port or through the Data port. Figure 2-8 shows sample programs reading the Status register contents through the Control port into the accumulator (A register) of an Am8080A/8085A system or the R0 register of an AmZ8002 system. It is assumed that the AmZ8002 system has a 16-bit data bus; since the status register is only eight bits wide, the high byte of register R0 is undefined.

The procedure for reading the Status register through the Control port is given in the following.

1. Establish a High on the C/$\overline{D}$ input.
2. Establish a Low on the $\overline{CS}$ input.
3. After the appropriate $\overline{CS}$ and C/$\overline{D}$ setup time (see Timing Diagram) make $\overline{RD}$ Low.
4. Sometime after $\overline{RD}$ goes Low, the Status register contents will appear on the data bus. These lines will contain the information as long as $\overline{RD}$ is Low. If the state of an OUT pin changes while $\overline{RD}$ is Low, this will be reflected by a change in the information on the data bus.
5. $\overline{RD}$ can be driven High to conclude the read operation after meeting the minimum $\overline{RD}$ pulse duration.
6. $\overline{CS}$ and C/$\overline{D}$ can change after meeting the appropriate hold time requirements (see Timing Diagram).

A new read or write operation to the Am9513 should not be attempted until the read recovery time is met (see Timing Diagram in Electrical Specification).

```
                        ;
                        ;       CODE TO READ STATUS REGISTER
                        ;
0113 DB12               IN      CMDPRT
                        ;
                        ;
                        PAGE
```

**a) 8080 Code**

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1    Page 3

0026                            %
0026                            %       CODE TO READ FROM STATUS REGISTER
0026                            %
0026    3B04 0012               IN      R0,CMDPRT;
002A                            %
002A                            %
002A                            EJECT;
```

**b) AmZ8000 Code**

**Figure 2-8. Reading the Status Register**

### Reading From the Data Port

The registers which can be read from the Data port are the Load, Hold and Counter Mode registers for Counters 1 through 5, the Alarm registers for Counters 1 and 2, the Master Mode register and the Status register. The Status register can also be read from the Control port. Reading the Status register with a 16-bit data bus interface will return undefined information on DB8-DB15.

The procedure for reading these registers is as follows:

1. Prior to performing the actual read operation, the Data Pointer should be set to point to the register to be read, as outlined in the "Setting the Data Pointer" section of this document. In cases where auto-sequencing of the Data Pointer is used, the Pointer has to be set only once to the first register in the sequence. When auto-sequencing is disabled, repetitive accesses can be made to the same register without reloading the Data Pointer each time. Special care must be taken to reset the Data Pointer after issuing a command other than "Load Data Pointer" to the Am9513 or when operating a counter in modes N, O, Q or R. See the "Prefetch Circuit" section of this document for elaboration.
2. Establish a Low on the C/$\overline{D}$ input.
3. Establish a Low on the $\overline{CS}$ input.
4. Establish a Low on $\overline{RD}$ after waiting for the appropriate $\overline{CS}$ and C/$\overline{D}$ setup time (see Timing Diagram).
5. Sometime after $\overline{RD}$ goes Low, the register contents will appear on the data bus. In both 8- and 16-bit bus modes the low register byte will appear on DB0-DB7. In addition, in 16-bit bus mode, the upper register byte will appear on the DB8-DB15. For 8-bit bus mode, pins DB8-DB15 are not driven by the Am9513. This information will remain stable as long as $\overline{RD}$ is Low. If the register value is changed during the read, the change will not be reflected by a change in the data being read, for the reasons outlined in the "Prefetch Circuit" section of this document.
6. $\overline{RD}$ can be driven High to conclude the read operation after meeting the minimum $\overline{RD}$ pulse duration.
7. $\overline{CS}$ and C/$\overline{D}$ can change after meeting appropriate hold time requirements (see Timing Diagram).
8. After waiting the minimum read recovery time (see Timing Diagram), a new read or write operation can be started. For 8-bit bus mode, steps 2 through 7 should be repeated to read out the high register byte on DB0-DB7. (If the Status register is being read in 8-bit mode, the two reads will return the Status register each time. In 16-bit mode, reads from the Status register return undefined data on DB8-DB15.) The user is not required to drive $\overline{CS}$ or C/$\overline{D}$ High between successive reads or writes, although this is permissible.

As described in the "Setting the Data Pointer" register section, the Am9513 service routines should disable interrupts during Data port register accesses if the service routine could be interrupted by another service routine requiring access to Data port registers.

Figure 2-9 shows sample programs for reading a Data port register. The Am8080A/8085A code reads the data in two byte reads (low byte first) and assembles it into the HL register pair. The AmZ8002 program assumes that a 16-bit data interface is being used and reads the data into register R0 in a single word read. This code can be substituted into the sample interrupt service routines in Figure 2-7 in the place marked "Code to Access Registers."

```
                        ;
                        ;       CODE TO READ FROM DATA PORT REG.
                        ;
0115 DB10               IN      DATAPRT
0117 6F                 MOV     L,A
0118 DB10               IN      DATAPRT
011A 67                 MOV     H,A
                        ;
                        ;
                        PAGE
```

**a) 8080 Code**

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 4

002A                            %
002A                            %       CODE TO READ FROM DATA PORT REG.
002A                            %
002A    3B24 0010               IN      R2,DATAPRT;
002E                            %
002E                            %
002E                            EJECT;
```

**b) AmZ8000 Code**

**Figure 2-9. Reading Through the Data Port**

### Writing to the Data Port

The registers which can be written to through the Data port are the Load, Hold and Counter Mode registers for Counters 1 through 5, the Alarm registers for Counters 1 and 2 and the Master Mode register. The procedure for writing to these registers is as follows:

1. Prior to performing the actual write operation, the Data Pointer should be set to point to the register to be written to, as outlined above in the "Setting the Data Pointer" section of this document. In cases where auto-sequencing of the Data Pointer is used, the Pointer has to be set only once to the first register in the sequence. When auto-sequencing is disabled, repetitive accesses can be made to the same register without reloading the Data pointer each time.
2. Establish the appropriate data on the DB0-DB7 lines (8-bit bus mode) or DB0-DB15 (16-bit bus mode). When using the 8-bit bus mode, data bus lines DB13-DB15 should be set High during the write operation and DB0-DB7 should be set to the lower data byte for the first write and to the upper data byte for the second write.
3. Establish a Low on the C/$\overline{D}$ input.
4. Establish a Low on the $\overline{CS}$ input.
5. Establish a Low on the $\overline{WR}$ input.
6. Drive $\overline{WR}$ High sometime after the minimum $\overline{WR}$ low pulse duration has been achieved, taking care the $\overline{CS}$, C/$\overline{D}$ and data setup times are met (see Timing Diagram).
7. After meeting the required $\overline{CS}$, C/$\overline{D}$ and data hold times, these signals can be changed (see Timing Diagram).
8. After meeting the write recovery time (see Timing Diagram) a new read or write operation can be performed. For the 8-bit bus mode, steps 2 through 7 should be repeated, this time placing the high data byte on pins DB0-DB7. The user is not required to drive $\overline{CS}$ or C/$\overline{D}$ High between successive reads or writes, although this is permissible.

As described in the "Setting the Data Pointer" section, Am9513 service routines should disable interrupts during Data port register accesses if the service routine could be interrupted by another service routine requiring access to the Data port registers.

Figure 2-10 shows sample programs for writing a 16-bit value to a Data port register. The Am8080A/8085A code loads the register by making two byte transfers (low byte first) to the Am9513 Data port. A 16-bit data bus interface is assumed for the AmZ8002 coding example; accordingly, a single word transfer can be used to load a register. This code can be substituted into the sample interrupt service routines in Figure 2-7 in the place marked "Code to Access Registers."

```
                                ;
                                ;       CODE TO WRITE TO DATA PORT REG.
                                ;
011B 7D                         MOV     A,L
011C D310                       OUT     DATAPRT
011E 7C                         MOV     A,H
011F D310                       OUT     DATAPRT
                                ;
                                ;
                                ;
0121                            END

A>
```

**a) 8080 Code**

```
AM9513_EXAMPLES                 MACRO8000 AmZ8000 Assembler   1.0.1     Page 5

002E                            %
002E                            %       CODE TO WRITE TO DATA PRT REG.
002E                            %
002E    3B26 0010               OUT     DATAPRT,R2;
0032                            %
0032                            %
0032                            %
0032                            END.
```

**b) AmZ8000 Code**

**Figure 2-10. Writing Through the Data Port**

# Chapter 3
# Concatenating Counters

## CONCATENATING COUNTERS

The Am9513 counters may be concatenated in a number of different ways. These may be conceptually broken down into count up and count down concatenation. Count up concatenation will typically be used to count events with a precision greater than 16 bits. Count down concatenation is typically used to generate output frequencies of high resolution.

To simplify concatenation, the Am9513 provides an internal TC signal from the low order counter which can be selected as a count source in the high order counter's Counter Mode register. Thus, although any two counters can be concatenated with external strapping, usually adjacent counters will be used to allow use of this internal TC signal.

In count up concatenation, both the high and low order counter's Load register should be cleared to 0. The low order counter will start counting up from 0 and increment through 9999. (BCD counting is assumed throughout this discussion, although binary counting may, of course, be used). On the next source edge the low order counter will go to TC and reload 0 from the Load register. The active-going TC edge will also increment the high order counter. The counters continue counting in this manner with the high order counter incrementing each time the low order counter reaches TC. In the examples which follow, Counters 1 and 2 will be used as the low order and high order counters respectively.

In the first up concatenation configuration, shown in Figure 3-1, the counters do not use external gating and therefore will free run. The high order counter should use the TC output of the low order counter as a source. The high order counter should count on rising source edges and should be programmed for "no gating." The above requirements can be met by specifying 00 (hex) in the upper byte of the high order counter's Mode register. The low order counter should be programmed to count repetitively. The required Mode register settings for Counters 1 and 2 are shown in the figure; "don't care" bits are marked "X." Note that if the internal TC signal is used to concatenate to the upper counter, no restrictions are placed on the programming of the low order counter's Output Control field. Conversely, if external strapping is used to concatenate the counter, the low order counter should have an "Active High TC" output mode selected. Up count concatenation may also be used with either level or edge gating. For level gating, the count source may either be externally gated with external logic, or the low order counter may be programmed for level gating, as shown in Figure 3-2. In either case, the high order counter should be programmed for "no gating." Recall that while in the TC state, the counters will count all source pulses issued to them, irrespective of their gating or arming status. This can introduce counting errors when level gating is used in up count concatenation. If the gate goes inactive while the low order counter is in TC, the low order counter will count the next source edge, which drives it out of TC. The counter will then stop counting until the gate goes active again. This effectively introduces a 1 count error into the accumulated count. The maximum error that can be introduced is one extra count each time the gate is applied. This worst case error will occur only if the gate is always applied when the low order counter is in the TC state. For many applications which use the gate infrequently, this small potential error is of no significance. Applications sensitive to small count errors or applications with many gate-on, gate-off cycles should use external gating logic to inhibit source pulses.

Edge gating functions can also be used in up count concatenation. An edge gating circuit with concatenated counters should function in a logically identical manner to a single edge-gated counter. In other words, after an edge is applied to the concatenated counters, they should count until both reach TC. A new edge should be required to repeat the cycle. Direct concatenation of two counters as was done for level gating up count

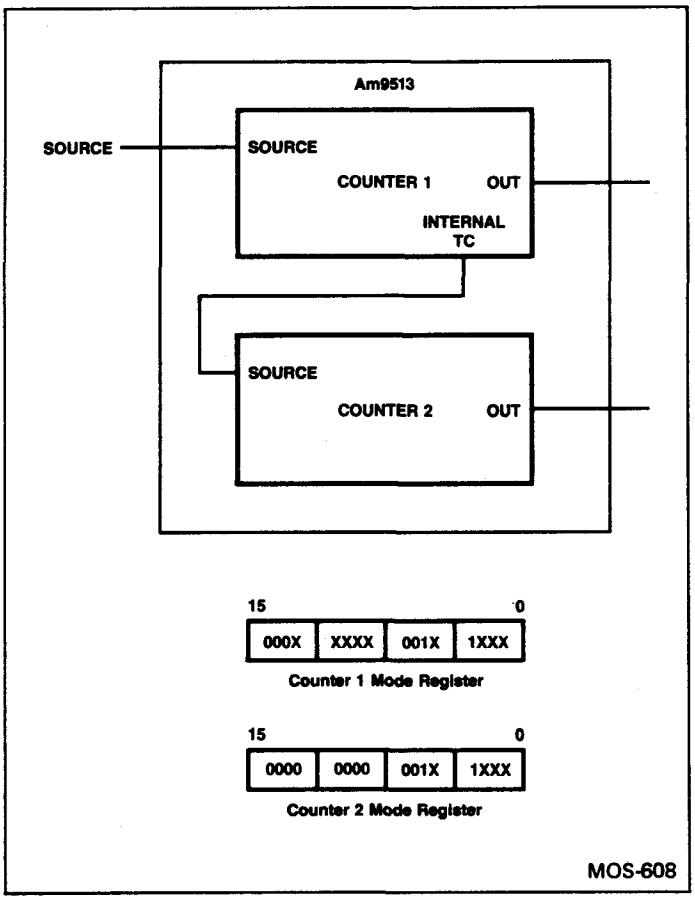



**Figure 3-1. Count Up Concatenation with No Gating**

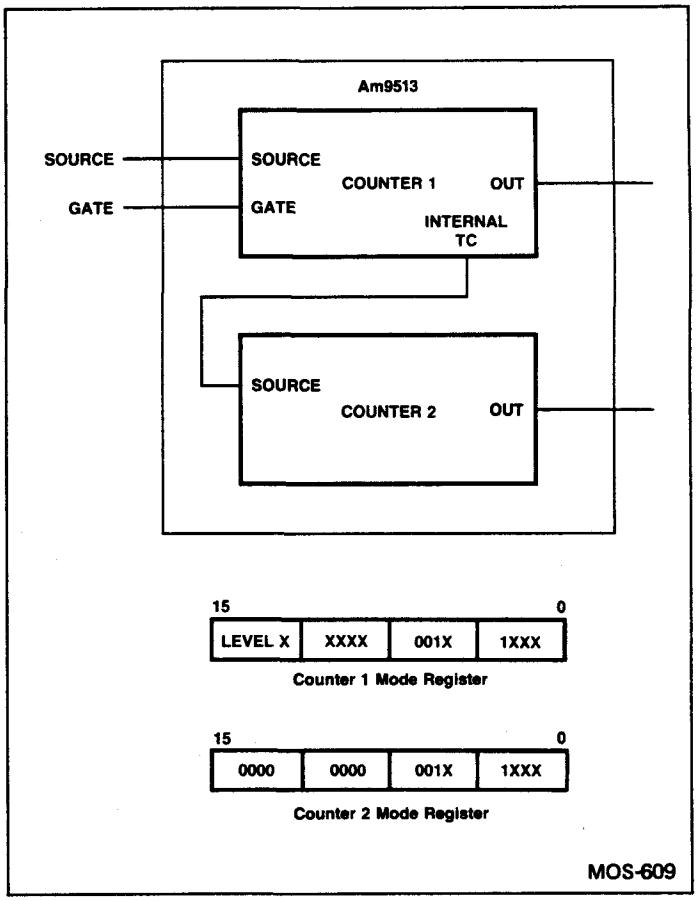



**Figure 3-2. Count Up Concatenation with Level Gating**

concatenation will not work. In such an arrangement, the low order counter, once triggered, will count to TC once and then stop, awaiting a new gate edge. This is unsatisfactory since we want the low order counter to continue counting until the high order counter reaches TC.

Figure 3-3 shows one method of concatenating counters for edge-triggered up counting. This method operates the counters in a similar arrangement to that used for level gating, with the requirement that each counter's output be programmed for an active-high TC pulse.

The external flip-flop is set by an external, synchronous gate signal. When both counters reach TC, the flip-flop is cleared. One potential problem exists with this scheme. Once the flip-flop clears, it will inhibit the low order counter's gate input. The low order counter will nevertheless count the next source edge, driving itself out of TC. However, the high order counter will remain in TC. When the next triggering gate edge is applied, the flip-flop will set, allowing counting to begin. When the low counter reaches its first TC, the rising TC edge will cause the high counter to leave TC. For a short period of time (the propagation delay of the high order counter from source to output), both TCs will again be active. This could potentially clear the flip-flop prematurely. To inhibit this, the source signal is added as an additional input to the NAND gate. If a relatively slow source is used with a high time greater than the total propagation delay from the source input of low order counter to the output of the high order counter, the source input on the NAND gate will inhibit clearing of the flip-flop during this transient.

The concatenation examples so far have assured that the counters are to "count repetitively," in the sense of counter mode register bit CM5. If "count once" operation is desired, in which the counters require an Arm command after each count cycle, different circuits are required.

When "no gating," "count once" operation is desired, the circuit in Figure 3-4 can be used. In this application, Counter 1 should be programmed for active-high level gating and Counter 2 should be programmed for a TC Toggled output. During counter initialization, the following set of commands should be used:

Initialize Counters' 1 and 2 Mode and Load registers
LOAD Counters 1 and 2
Clear Counter 2 output
ARM Counters 1 and 2.

The counters are now ready to count, but since Counter 2's output is low, Counter 1's gate will inhibit counting. To start counter operation, use the "Set Counter 2's output" command. The counters will then count applied source pulses until Counter 2 reaches TC and toggles its output, inhibiting Counter 1's gate. It can be seen that in this application, the "Set Counter 2's output" behaves as an ARM command. It is important that the counting rate be low enough to ensure that Counter 1's gate will not go inactive in close proximity to a source edge. High speed applications using a Counter 1 source period less than the propagation delay from Counter 1's source to Counter 2's output should use a flip-flop to synchronize Counter 2's output to Counter 1's source in order to meet timing parameters TGVEH and TEHGV in the Am9513 data sheet. High-speed applications will end the count cycle with a value slightly larger than 1 in Counter 1.

To add level gating to this "count once" feature simply involves the addition of an AND function before Counter 1's gate input. Now Counter 1 will be inhibited whenever Counter 2 toggles its output or whenever the external gate is driven low. Note that this circuit assumes the externally applied gate is synchronous to the count source; asynchronous gating signals should be synchronized with a flip-flop.

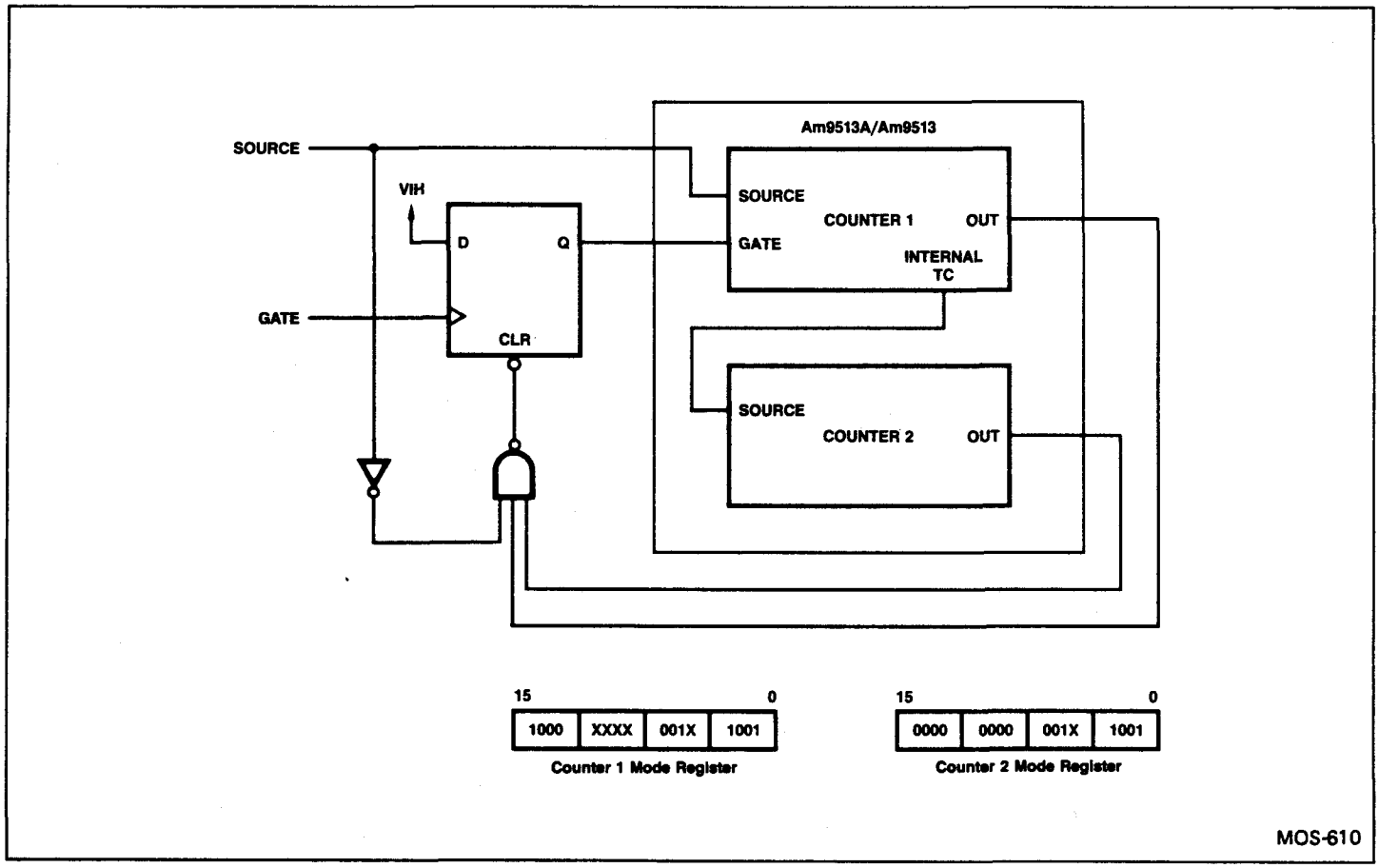



**Figure 3-3. Count Up Concatenation with Edge Gating**

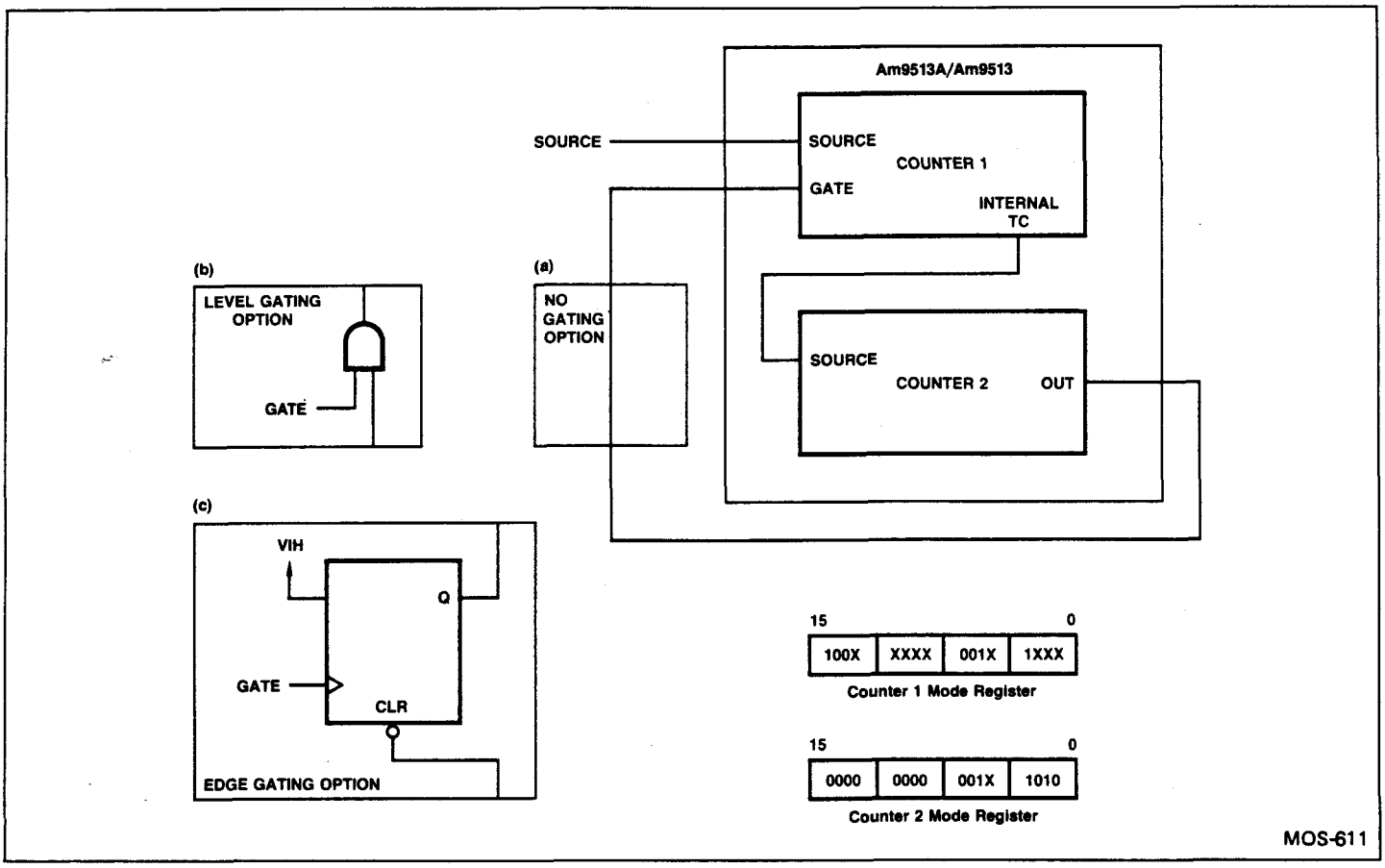



**Figure 3-4. Count Up Concatenation with Count Once Feature**

The final case of concatenated up counting comprises edge gating with the "count once" feature. This is achieved through a simple variation of the level gating configuration. An external gate signal sets the flip-flop and enables counting providing Counter 2's output is set. When Counter 2 reaches TC, its output will toggle (i.e., clear) and the flip-flop will clear, inhibiting further counting. To restart the counter in this configuration, the "Set Counter 2 output" command should be issued and a new gate edge should be applied in the order. As in the previous cases, the applied gate edge should be synchronous to the Counter 1 source.

In order to analyze down concatenation, it is useful to separately analyze the sequences followed for the high order and low order counters. Figure 3-5 shows a typical count down concatenation sequence, with the high order and low order count sequences labelled. The high order counter simply decrements from some initial value L until TC is reached. (In the following discussion and figures, L and H are used to represent the Load and Hold register contents respectively; K and N are used to represent arbitrary count values.) It is then reloaded with L and repeats the sequence. Note that the high order counter, in general, will never count to 0, since TC is generated by the source edge occurring while the counter contains 1 and TC reloads the initial value L. The count sequence is thus L, (L−1), . . . , 2, 1, L, (L−1), (L−2), . . . , 2, 1, L. The low order counter starts from some initial value H and counts down to TC. This TC output will be used to decrement the high order counter by 1. The low order counter is now reloaded with 0 and counts down through (assuming BCD counting) 9999 to 1. This sequence of reloading 0 and counting down to the next

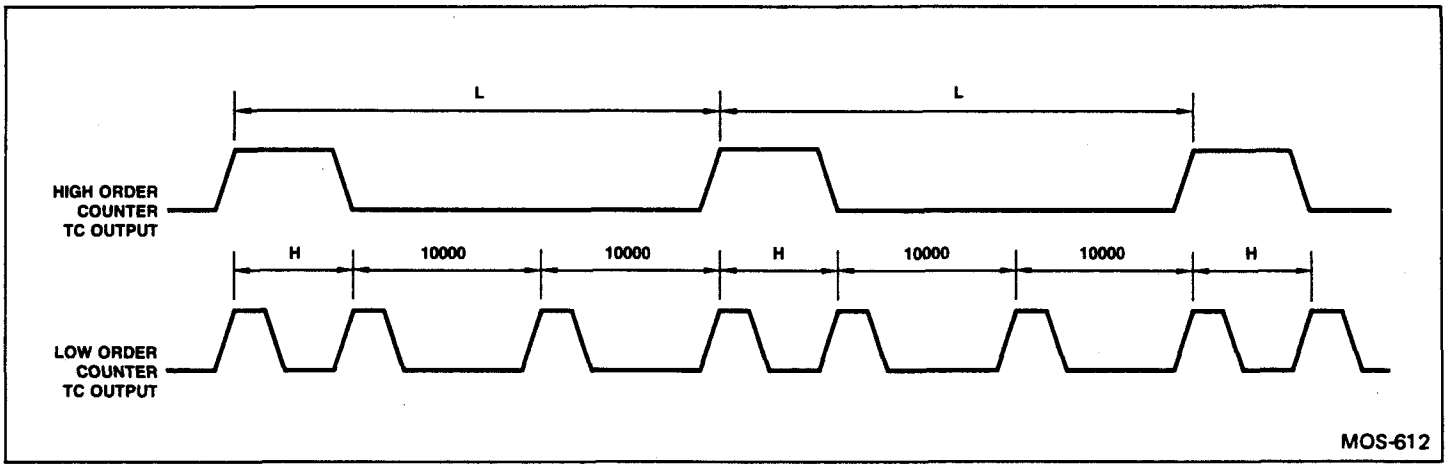



**Figure 3-5. Conceptual Sequence for Count Down Concatenation**

TC will be repeated by the low order counter until the high order counter has decremented to 1. On the next low order TC, the high order counter is driven to TC and reloads L. The low order counter should reload H, rather than 0, and repeat the complete count cycle. It can be seen that an important characteristic of the low order counter is that it reloads H once for each high order TC, and reloads 0 otherwise. This need for the low order counter to selectively reload 0 or H differs from up concatenation where the low order counter is always reloaded with the same value (0).

Figure 3-6 ties the above considerations together in a count repetitively, no gating, count down concatenation example. The low order counter is operated in Mode V, in which the gate is used

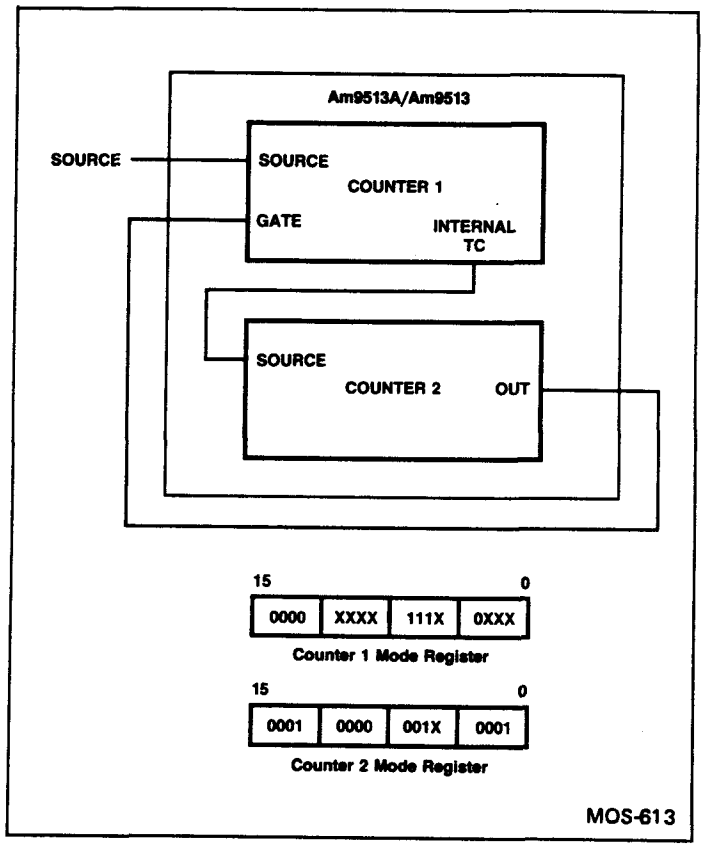



**Figure 3-6. Count Down Concatenation**

to select either the Load or Hold register as a reload source. The high order counter is operated in Mode D, with an active-high TC output selected in order to properly drive the low order counter's gate. In addition, the high order counter should be programmed to count on falling edges of the low order counter's internal TC output. Figure 3-7 shows timing waveforms generated by this concatenation configuration. Note that the count sequence generated never has 0 in the upper counter (disregarding the special case where L = 0). This means that the value stored in the high order counter should be biased by adding 1 in order to generate the correct divider ratio. For example, to divide by 39264178 (BCD), the high order counter's Load register should be set to 3926 + 1 = 3927 and the low order counter's Hold register should be set to 4178. The low order counter's Load register should be set to 0 to ensure proper count value rollover. Also note the unusual count sequence on the TC before the low order counter reloads from the Load register. For the above example of dividing by 39264178 the counters will count 00010002, 00010001, 00010000, 39279999, 39279998, . . . , 39270002, 39270001, 39274178, 39264177, 39264176, 39264175, . . . rather than 00010002, 00010001, 00010000, 39274178, 39274177, . . . , 39270002, 39270001, 39270000, 39269999, 39269998, 39269997, . . . .

In some applications it may be desirable to level or edge gate with down concatenation. Because the low order counter uses the gate to select the reload source, the gate input cannot be used to start and stop counting in the low order counter. Accordingly, external gating logic must be used. Figure 3-8 shows the connections required for count down concatenations with level gating. Level gating is achieved by inhibiting source pulses when the gate goes inactive.

Edge gating, shown in Figure 3-9, uses an external gate signal to set an enabling flip-flop. The enabling flip-flop is cleared when both counters reach TC. The delay flip-flop ensures that one additional count occurs after both counters reach TC in order to drive the low order counter out of TC, thereby deactivating the enabling flip-flop's clear input. Note that the counters stop at an unusual point in the count sequence, ((L−1), (H−1) in Figure 3-7 or 3926 4177 for the earlier example) but this is not important

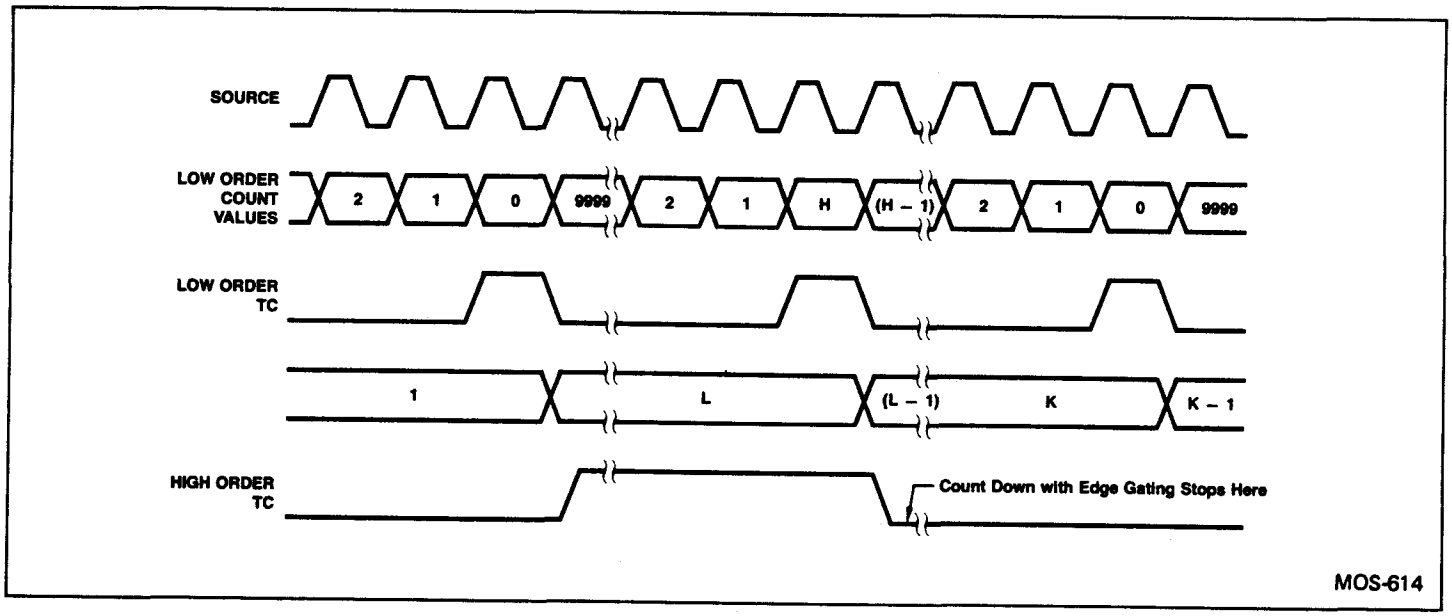



**Figure 3-7. Timing Waveforms for Am9513 Count Down Concatenation**

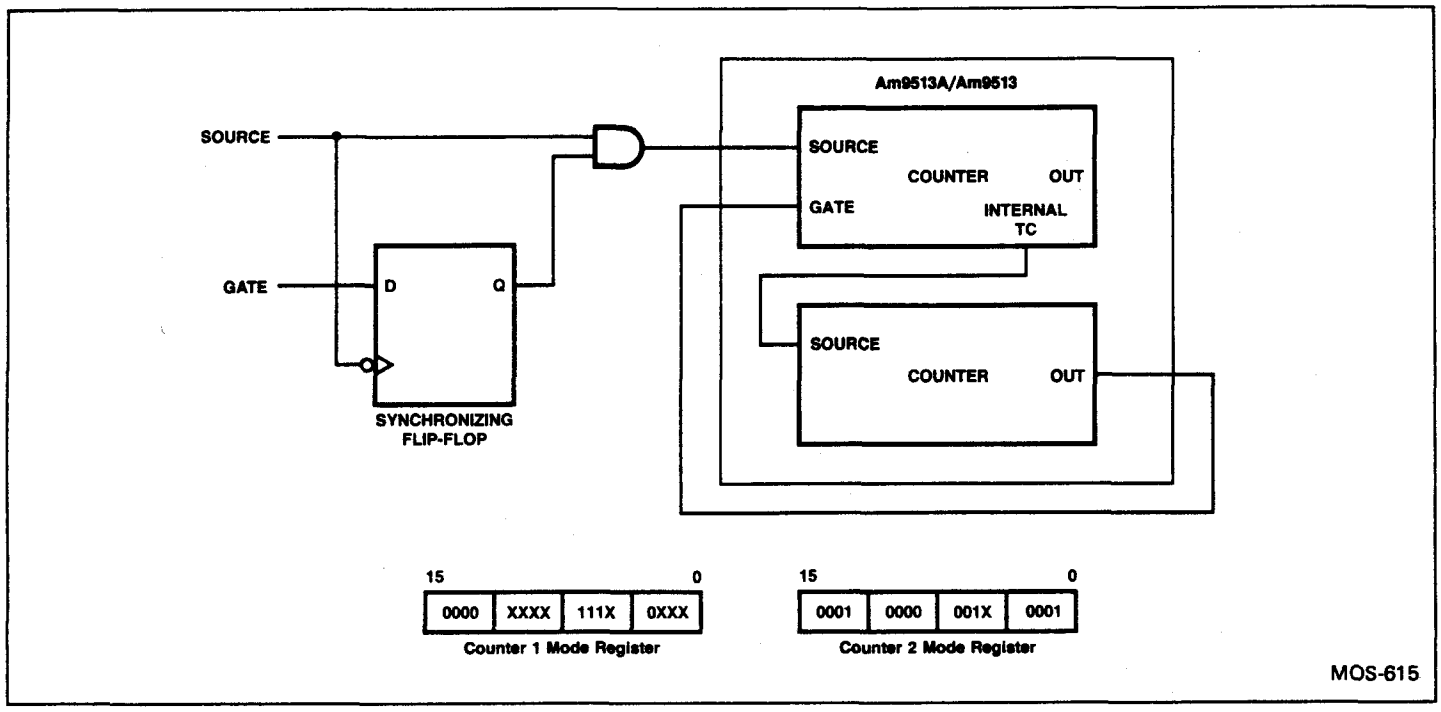



**Figure 3-8. Count Down Concatenation with Level Gating**

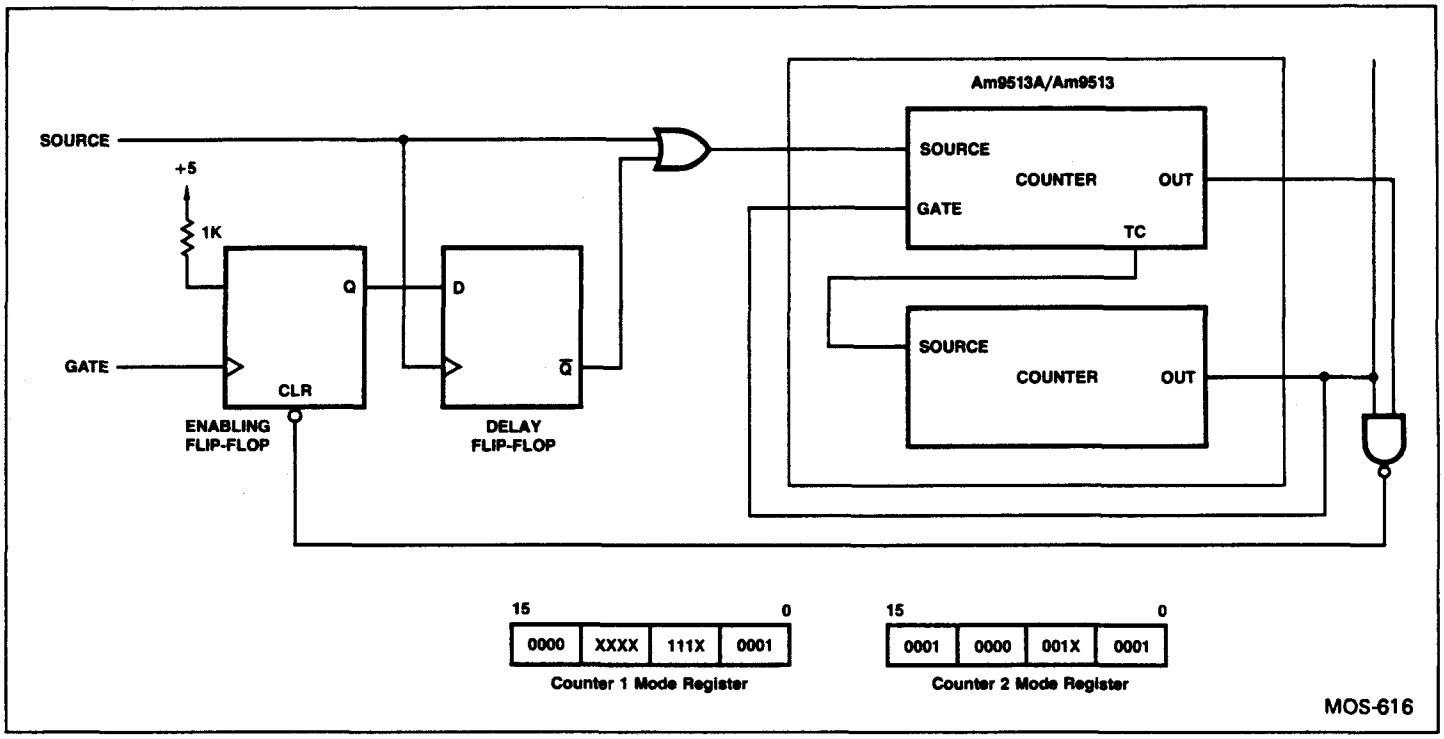



**Figure 3-9. Count Down Concatenation with Edge Gating**

since the timeout duration remains constant (at (L−1), H for Figure 3-7 and 3926 4178 for the earlier example). To ensure that the counters' first timing cycle has the same timeout duration as subsequent timing cycles, it is important that the high and low order counters be initialized to (L−1) and (H−1) respectively prior to the first timing cycle. Note that if the Counter 1 source period is less than the propagation delay from Counter 1's source through Counter 2's output, through the two flip-flops to the OR gate, then the low order counter's contents at the end of a count cycle may be offset by a few counts. In such cases, the value used to initialize the counters should be similarly offset.

The previous count down concatenation examples have assumed the counters are to count repetitively. To add count once capability to a count down configuration, the high order counter should be programmed to generate a TC Toggled output waveform. This output should be used to gate source pulses through an AND gate into the low order counter, as shown in Figure 3-10. The count cycle will now appear as shown in Figure 3-11. Note that when the counters stop, the high order counter's output will be low and the low order counter's contents will be 9999. To reset the counters for another timing cycle, a LOAD command should be issued to the low order counter, which will

# Intel 82C55A Programmable Peripheral Interface
# Data Sheet Reprint

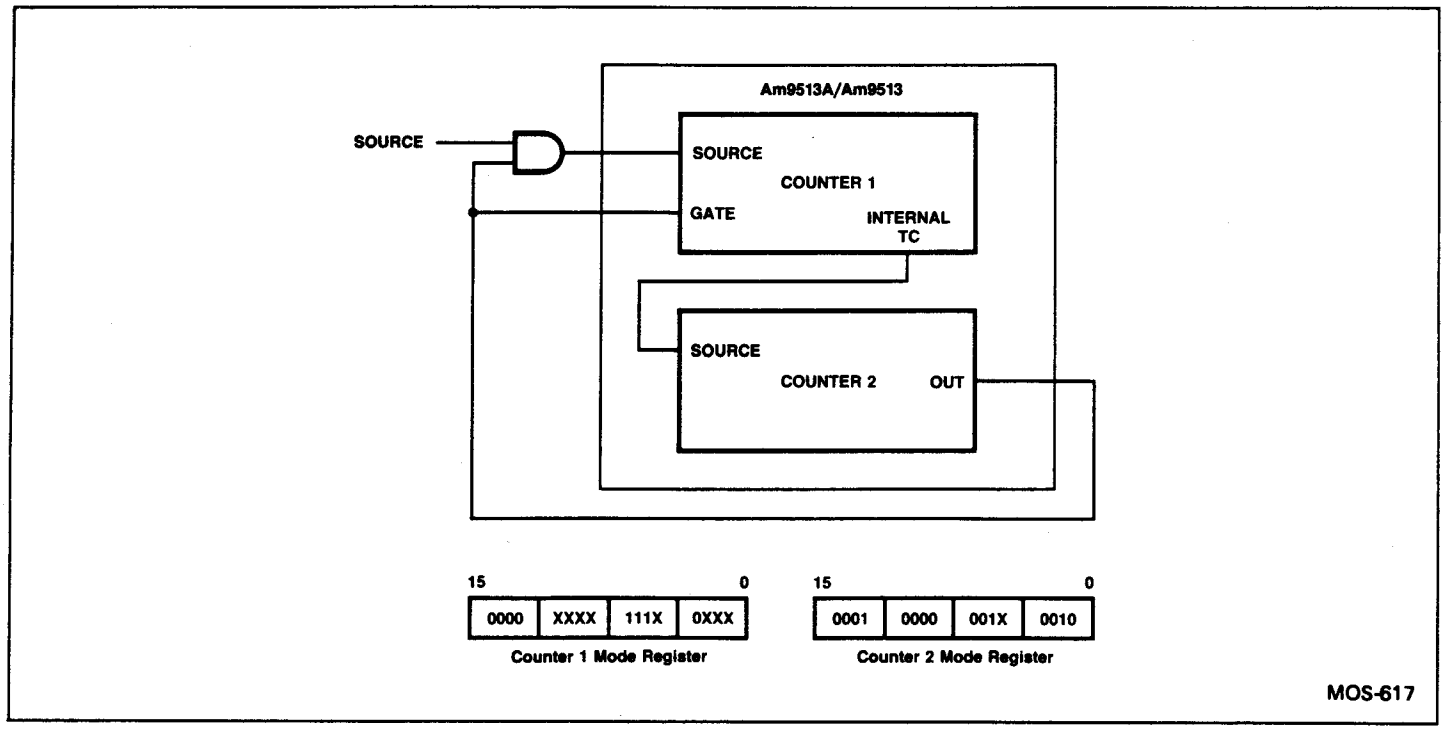



**Figure 3-10. Count Down Concatenation with Count Once Feature**

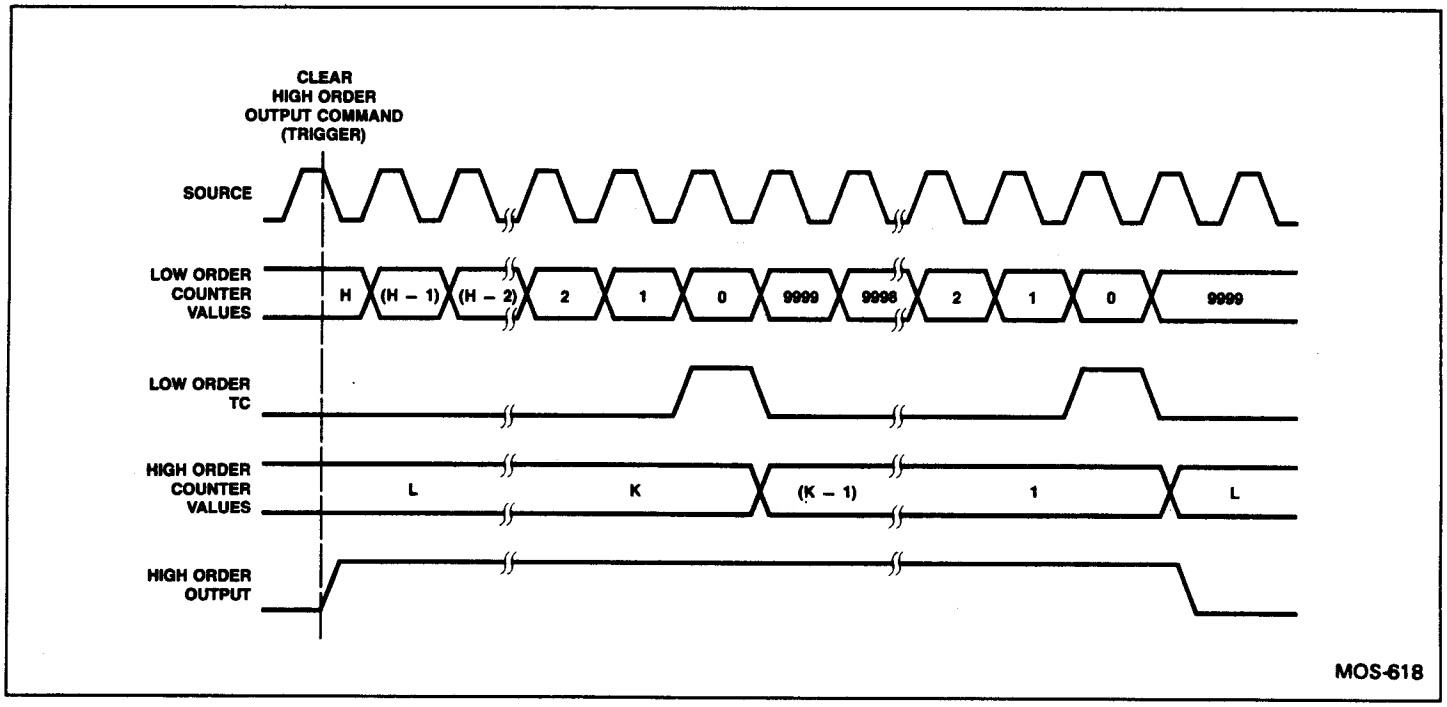



**Figure 3-11. Timing Waveforms for Count Down Concatenation with Count Once Feature**

reload from the Hold register. The output of the high order counter can now be set to enable counting. Level gating can be added to count-once, count down concatenation by using a 3-input AND gate and driving the third input with an external level gate signal. Count-once, count down concatenation with edge gating can be achieved with the circuit shown in Figure 3-12. The flip-flop is set by an external synchronous gate edge; it is cleared at the end of the count cycle when Counter 2's TC Toggled output goes low.

The concatenation examples presented so far have used two counters to create a 32-bit effective count length. These configurations can be extrapolated to concatenate 3 or more counters to any desired length. Other concatenation variations adventuresome users may wish to investigate are those that use the Alarm registers on Counter 1 and 2 to generate unusual count sequences. Since these Alarm register configurations usually add much complexity for only a limited increase in functionality they are not discussed in this manual.

### Saving Concatenated Count Values

The contents of concatenated counters may be read by issuing a SAVE command to the appropriate counters, which will transfer the current counter contents into the counter's Hold registers.

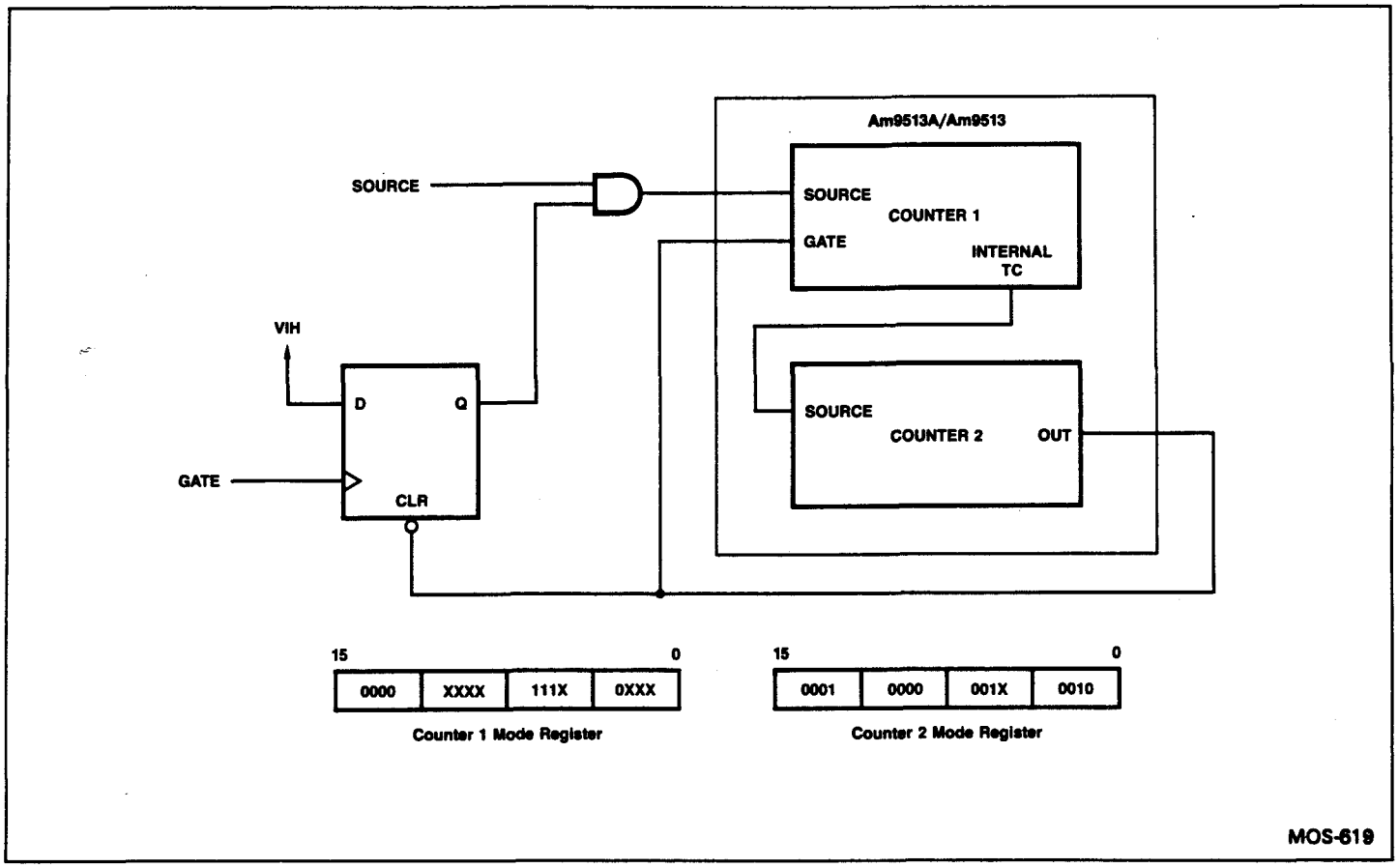



**Figure 3-12. Count Down Concatenation with Edge Gating and Count Once Feature**

(Since in count down concatenation the Hold register is used to generate the count sequence, in many such applications it may not be feasible to save the low-order counter.) Because the count ripples between concatenated counters, the possibility exists that a SAVE command will be issued after the low order counter increments/decrements but before the carry/borrow ripples through to the high order counter, resulting in an incorrect value being saved in the high order counter's Hold register. The user can protect against this by examining the contents of the low order counter's Hold register immediately after issuing the SAVE command. If the Hold register is equal to the value that would have been expected immediately following generation of a carry/borrow signal, this indicates that the high order value saved is suspect. A new SAVE command should therefore be issued to the high order counter to save a correct count. By the time the low order Hold register contents are read and tested, and a new SAVE command is issued, the high order counter's contents will be stable. The "Time-of-Day" chapter discusses these considerations with respect to Time-of-Day accumulation, and includes a representative software listing.

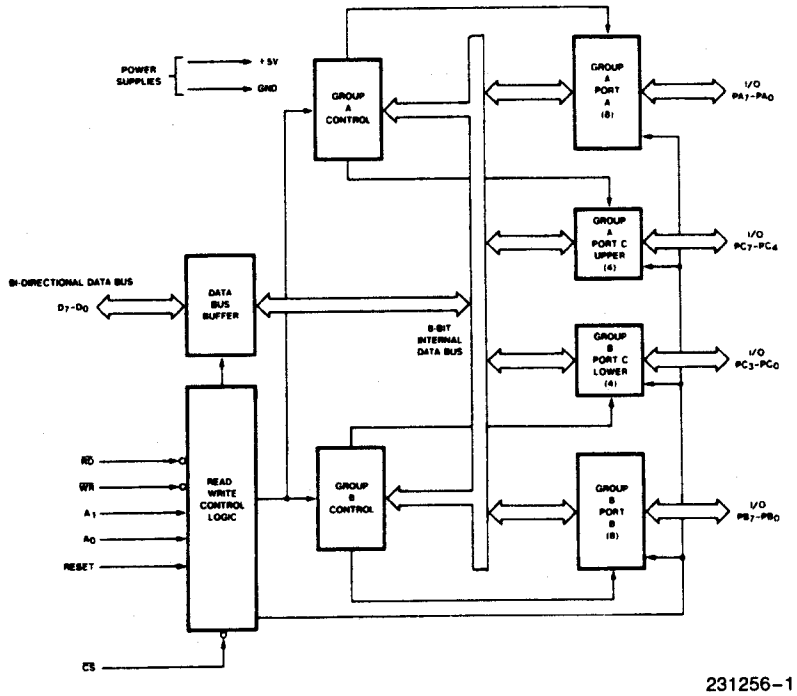
# 82C55A
# CHMOS PROGRAMMABLE PERIPHERAL INTERFACE

- **Compatible with all Intel and Most Other Microprocessors**
- **High Speed, "Zero Wait State" Operation with 8 MHz 8086/88 and 80186/188**
- **24 Programmable I/O Pins**
- **Low Power CHMOS**
- **Completely TTL Compatible**
- **Control Word Read-Back Capability**
- **Direct Bit Set/Reset Capability**
- **2.5 mA DC Drive Capability on all I/O Port Outputs**
- **Available in 40-Pin DIP and 44-Pin PLCC**
- **Available in EXPRESS**
  - **— Standard Temperature Range**
  - **— Extended Temperature Range**

The Intel 82C55A is a high-performance, CHMOS version of the industry standard 8255A general purpose programmable I/O device which is designed for use with all Intel and most other microprocessors. It provides 24 I/O pins which may be individually programmed in 2 groups of 12 and used in 3 major modes of operation. The 82C55A is pin compatible with the NMOS 8255A and 8255A-5.

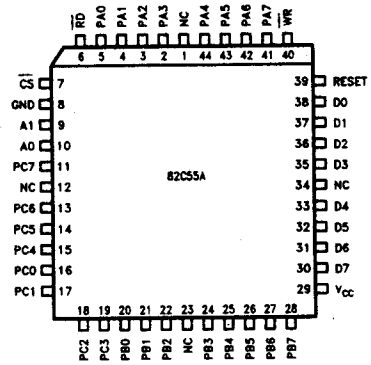
In MODE 0, each group of 12 I/O pins may be programmed in sets of 4 and 8 to be inputs or outputs. In MODE 1, each group may be programmed to have 8 lines of input or output. 3 of the remaining 4 pins are used for handshaking and interrupt control signals. MODE 2 is a strobed bi-directional bus configuration.

The 82C55A is fabricated on Intel's advanced CHMOS III technology which provides low power consumption with performance equal to or greater than the equivalent NMOS product. The 82C55A is available in 40-pin DIP and 44-pin plastic leaded chip carrier (PLCC) packages.

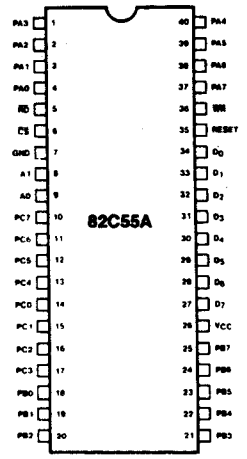



231256-1

**Figure 1. 82C55A Block Diagram**




231256-31




231256-2

**Figure 2. 82C55A Pinout**

Diagrams are for pin reference only. Package sizes are not to scale.



September 1987
Order Number: 231256-004

**Table 1. Pin Description**

| Symbol | Pin Number Dip | Pin Number PLCC | Type | Name and Function |
|---|---|---|---|---|
| $PA_{3-0}$ | 1-4 | 2-5 | I/O | **PORT A, PINS 0-3:** Lower nibble of an 8-bit data output latch/ buffer and an 8-bit data input latch. |
| $\overline{RD}$ | 5 | 6 | I | **READ CONTROL:** This input is low during CPU read operations. |
| $\overline{CS}$ | 6 | 7 | I | **CHIP SELECT:** A low on this input enables the 82C55A to respond to $\overline{RD}$ and $\overline{WR}$ signals. $\overline{RD}$ and WR are ignored otherwise. |
| GND | 7 | 8 | | **System Ground** |
| $A_{1-0}$ | 8-9 | 9-10 | I | **ADDRESS:** These input signals, in conjunction $\overline{RD}$ and $\overline{WR}$, control the selection of one of the three ports or the control word registers. |

| $A_1$ | $A_0$ | $\overline{RD}$ | $\overline{WR}$ | $\overline{CS}$ | Input Operation (Read) |
|---|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 | Port A - Data Bus |
| 0 | 1 | 0 | 1 | 0 | Port B - Data Bus |
| 1 | 0 | 0 | 1 | 0 | Port C - Data Bus |
| 1 | 1 | 0 | 1 | 0 | Control Word - Data Bus |
| | | | | | **Output Operation (Write)** |
| 0 | 0 | 1 | 0 | 0 | Data Bus - Port A |
| 0 | 1 | 1 | 0 | 0 | Data Bus - Port B |
| 1 | 0 | 1 | 0 | 0 | Data Bus - Port C |
| 1 | 1 | 1 | 0 | 0 | Data Bus - Control |
| | | | | | **Disable Function** |
| X | X | X | X | 1 | Data Bus - 3 - State |
| X | X | 1 | 1 | 0 | Data Bus - 3 - State |

| | | | | |
|---|---|---|---|---|
| $PC_{7-4}$ | 10-13 | 11,13-15 | I/O | **PORT C, PINS 4-7:** Upper nibble of an 8-bit data output latch/ buffer and an 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. |
| $PC_{0-3}$ | 14-17 | 16-19 | I/O | **PORT C, PINS 0-3:** Lower nibble of Port C. |
| $PB_{0-7}$ | 18-25 | 20-22, 24-28 | I/O | **PORT B, PINS 0-7:** An 8-bit data output latch/buffer and an 8-bit data input buffer. |
| $V_{CC}$ | 26 | 29 | | **SYSTEM POWER:** + 5V Power Supply. |
| $D_{7-0}$ | 27-34 | 30-33, 35-38 | I/O | **DATA BUS:** Bi-directional, tri-state data bus lines, connected to system data bus. |
| RESET | 35 | 39 | I | **RESET:** A high on this input clears the control register and all ports are set to the input mode. |
| $\overline{WR}$ | 36 | 40 | I | **WRITE CONTROL:** This input is low during CPU write operations. |
| $PA_{7-4}$ | 37-40 | 41-44 | I/O | **PORT A, PINS 4-7:** Upper nibble of an 8-bit data output latch/ buffer and an 8-bit data input latch. |
| NC | | 1, 12, 23, 34 | | No Connect |

## 82C55A FUNCTIONAL DESCRIPTION

### General

The 82C55A is a programmable peripheral interface device designed for use in Intel microcomputer systems. Its function is that of a general purpose I/O component to interface peripheral equipment to the microcomputer system bus. The functional configuration of the 82C55A is programmed by the system software so that normally no external logic is necessary to interface peripheral devices or structures.

### Data Bus Buffer

This 3-state bidirectional 8-bit buffer is used to interface the 82C55A to the system data bus. Data is transmitted or received by the buffer upon execution of input or output instructions by the CPU. Control words and status information are also transferred through the data bus buffer.

### Read/Write and Control Logic

The function of this block is to manage all of the internal and external transfers of both Data and Control or Status words. It accepts inputs from the CPU Address and Control busses and in turn, issues commands to both of the Control Groups.

### Group A and Group B Controls

The functional configuration of each port is programmed by the systems software. In essence, the CPU "outputs" a control word to the 82C55A. The control word contains information such as "mode", "bit set", "bit reset", etc., that initializes the functional configuration of the 82C55A.

Each of the Control blocks (Group A and Group B) accepts "commands" from the Read/Write Control Logic, receives "control words" from the internal data bus and issues the proper commands to its associated ports.

Control Group A - Port A and Port C upper (C7–C4)
Control Group B - Port B and Port C lower (C3–C0)

The control word register can be both written and read as shown in the address decode table in the pin descriptions. Figure 6 shows the control word format for both Read and Write operations. When the control word is read, bit D7 will always be a logic "1", as this implies control word mode information.

### Ports A, B, and C

The 82C55A contains three 8-bit ports (A, B, and C). All can be configured in a wide variety of functional characteristics by the system software but each has its own special features or "personality" to further enhance the power and flexibility of the 82C55A.

**Port A.** One 8-bit data output latch/buffer and one 8-bit input latch buffer. Both "pull-up" and "pull-down" bus hold devices are present on Port A.

**Port B.** One 8-bit data input/output latch/buffer. Only "pull-up" bus hold devices are present on Port B.

**Port C.** One 8-bit data output latch/buffer and one 8-bit data input buffer (no latch for input). This port can be divided into two 4-bit ports under the mode control. Each 4-bit port contains a 4-bit latch and it can be used for the control signal outputs and status signal inputs in conjunction with ports A and B. Only "pull-up" bus hold devices are present on Port C.

See Figure 4 for the bus-hold circuit configuration for Port A, B, and C.

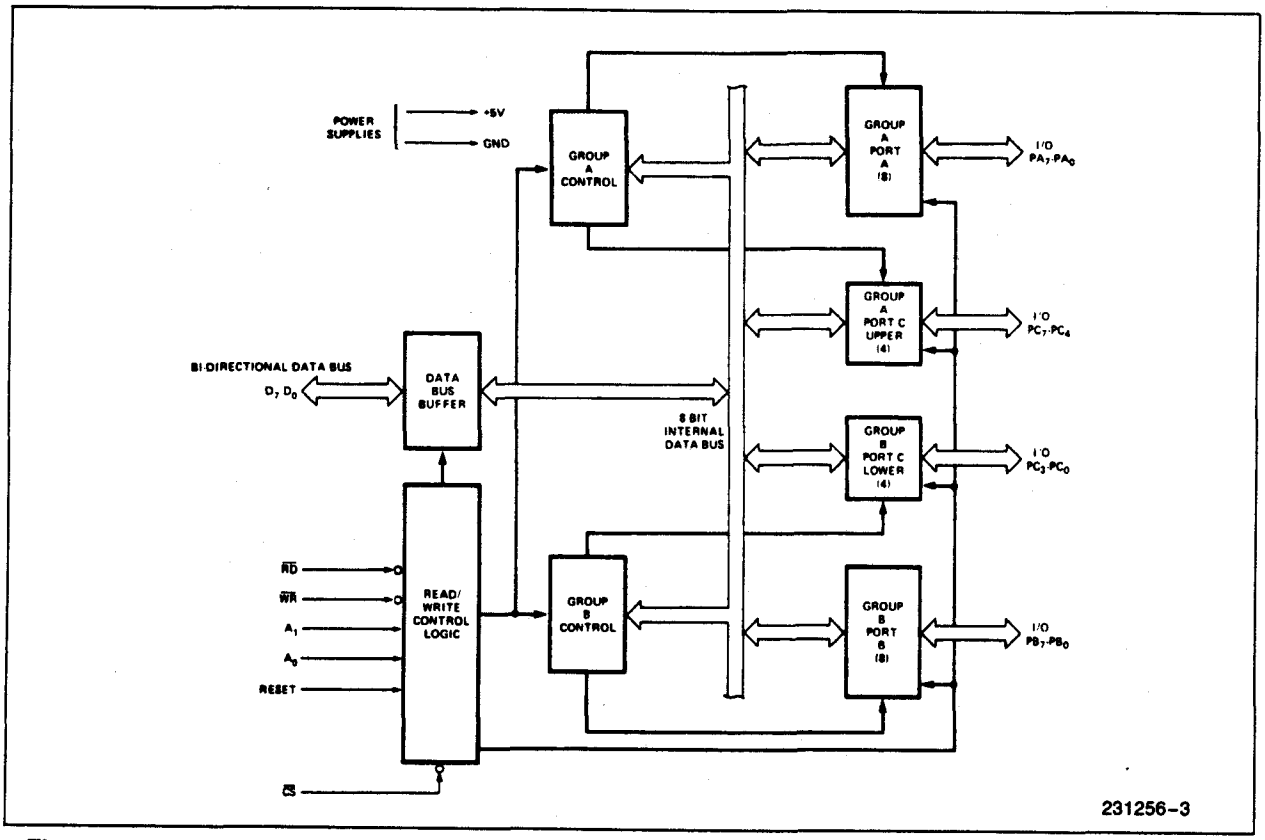
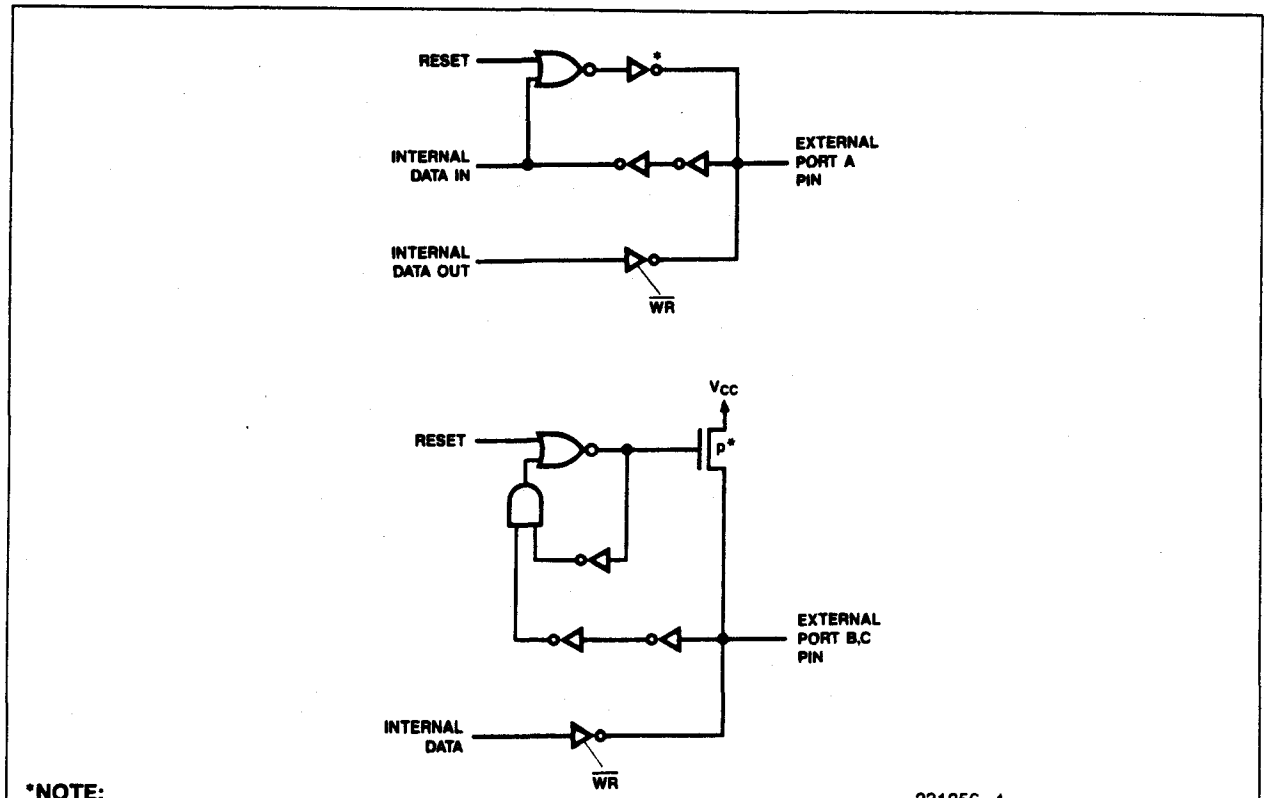



**Figure 3. 82C55A Block Diagram Showing Data Bus Buffer and Read/Write Control Logic Functions**




***NOTE:**
Port pins loaded with more than 20 pF capacitance may not have their logic level guaranteed following a hardware reset.

**Figure 4. Port A, B, C, Bus-hold Configuration**

## 82C55A OPERATIONAL DESCRIPTION

### Mode Selection

There are three basic modes of operation that can be selected by the system software:

Mode 0 — Basic input/output
Mode 1 — Strobed Input/output
Mode 2 — Bi-directional Bus

When the reset input goes "high" all ports will be set to the input mode with all 24 port lines held at a logic "one" level by the internal bus hold devices (see Figure 4 Note). After the reset is removed the 82C55A can remain in the input mode with no additional initialization required. This eliminates the need for pullup or pulldown devices in "all CMOS" designs. During the execution of the system program, any of the other modes may be selected by using a single output instruction. This allows a single 82C55A to service a variety of peripheral devices with a simple software maintenance routine.

The modes for Port A and Port B can be separately defined, while Port C is divided into two portions as required by the Port A and Port B definitions. All of the output registers, including the status flip-flops, will be reset whenever the mode is changed. Modes may be combined so that their functional definition can be "tailored" to almost any I/O structure. For instance; Group B can be programmed in Mode 0 to monitor simple switch closings or display computational results, Group A could be programmed in Mode 1 to monitor a keyboard or tape reader on an interrupt-driven basis.

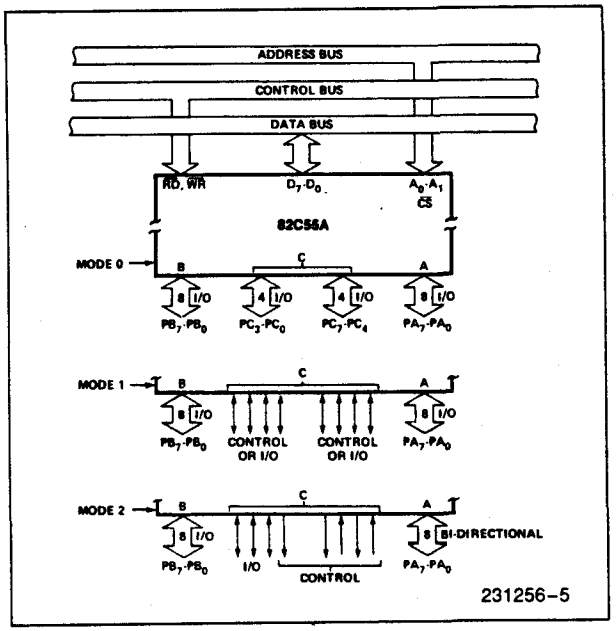



**Figure 5. Basic Mode Definitions and Bus Interface**

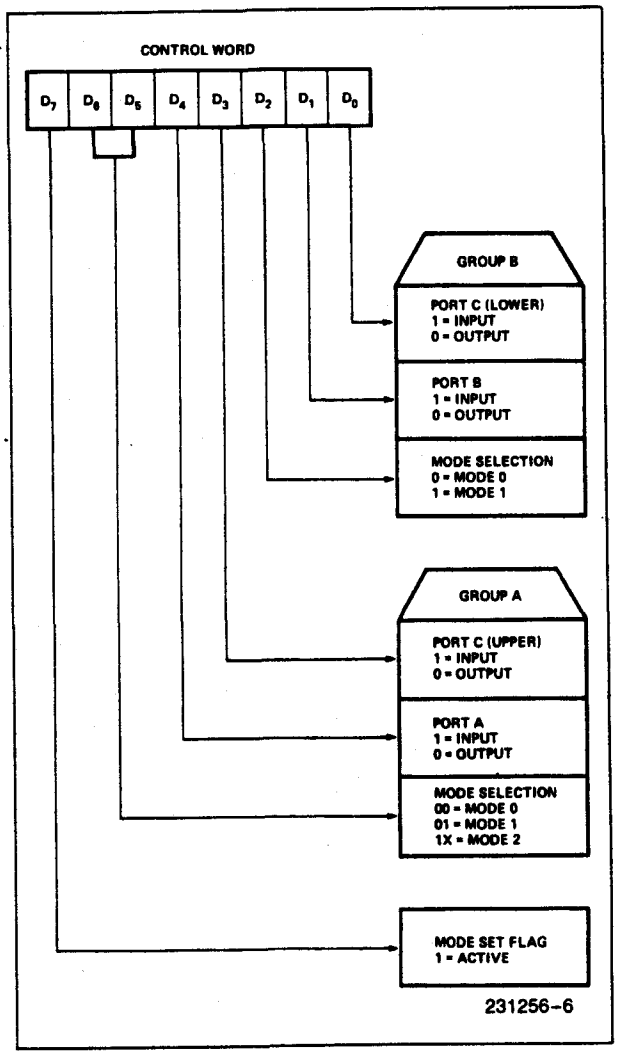



**Figure 6. Mode Definition Format**

The mode definitions and possible mode combinations may seem confusing at first but after a cursory review of the complete device operation a simple, logical I/O approach will surface. The design of the 82C55A has taken into account things such as efficient PC board layout, control signal definition vs PC layout and complete functional flexibility to support almost any peripheral device with no external logic. Such design represents the maximum use of the available pins.

### Single Bit Set/Reset Feature

Any of the eight bits of Port C can be Set or Reset using a single OUTput instruction. This feature reduces software requirements in Control-based applications.

When Port C is being used as status/control for Port A or B, these bits can be set or reset by using the Bit Set/Reset operation just as if they were data output ports.

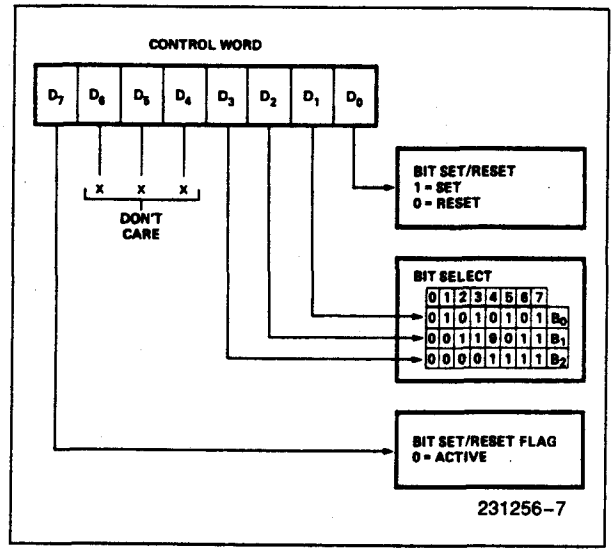



**Figure 7. Bit Set/Reset Format**

### Interrupt Control Functions

When the 82C55A is programmed to operate in mode 1 or mode 2, control signals are provided that can be used as interrupt request inputs to the CPU. The interrupt request signals, generated from port C, can be inhibited or enabled by setting or resetting the associated INTE flip-flop, using the bit set/reset function of port C.

This function allows the Programmer to disallow or allow a specific I/O device to interrupt the CPU without affecting any other device in the interrupt structure.

INTE flip-flop definition:

(BIT-SET)—INTE is SET—Interrupt enable
(BIT-RESET)—INTE is RESET—Interrupt disable

**Note:**
All Mask flip-flops are automatically reset during mode selection and device Reset.

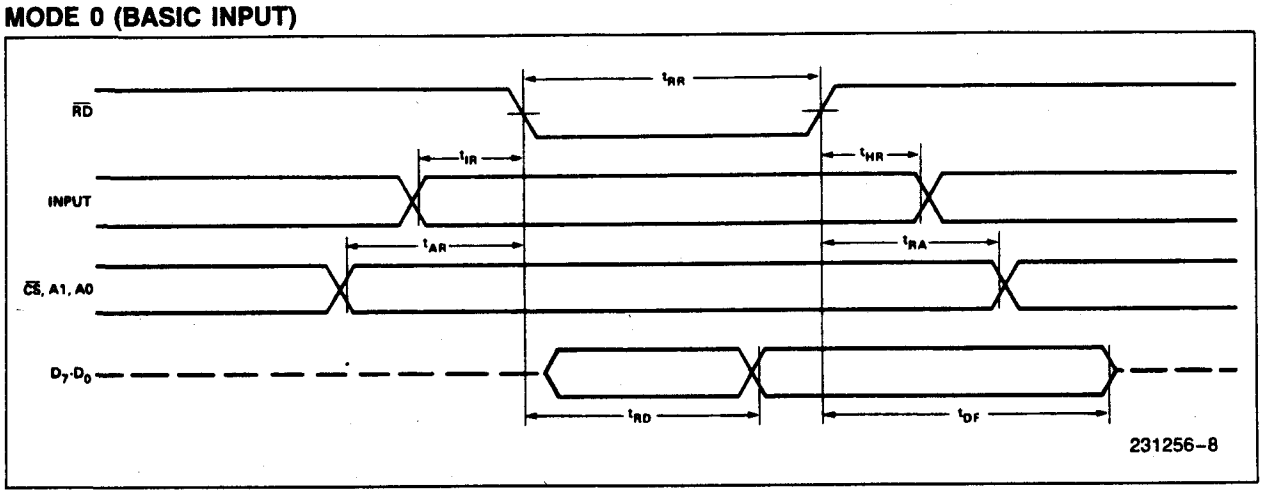
### Operating Modes

**Mode 0 (Basic Input/Output).** This functional configuration provides simple input and output operations for each of the three ports. No "handshaking" is required, data is simply written to or read from a specified port.

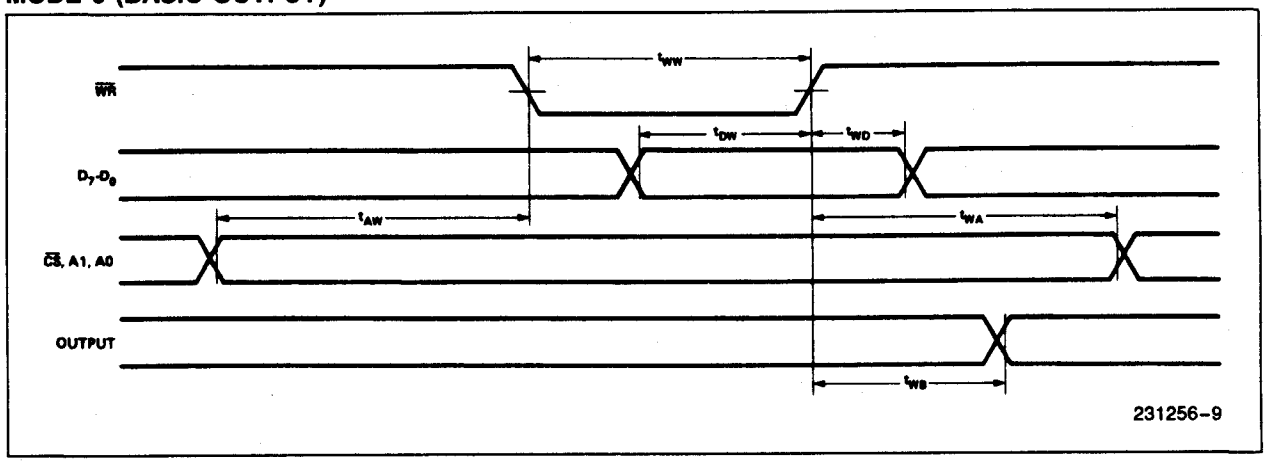
Mode 0 Basic Functional Definitions:

- Two 8-bit ports and two 4-bit ports.
- Any port can be input or output.
- Outputs are latched.
- Inputs are not latched.
- 16 different Input/Output configurations are possible in this Mode.

**MODE 0 (BASIC INPUT)**



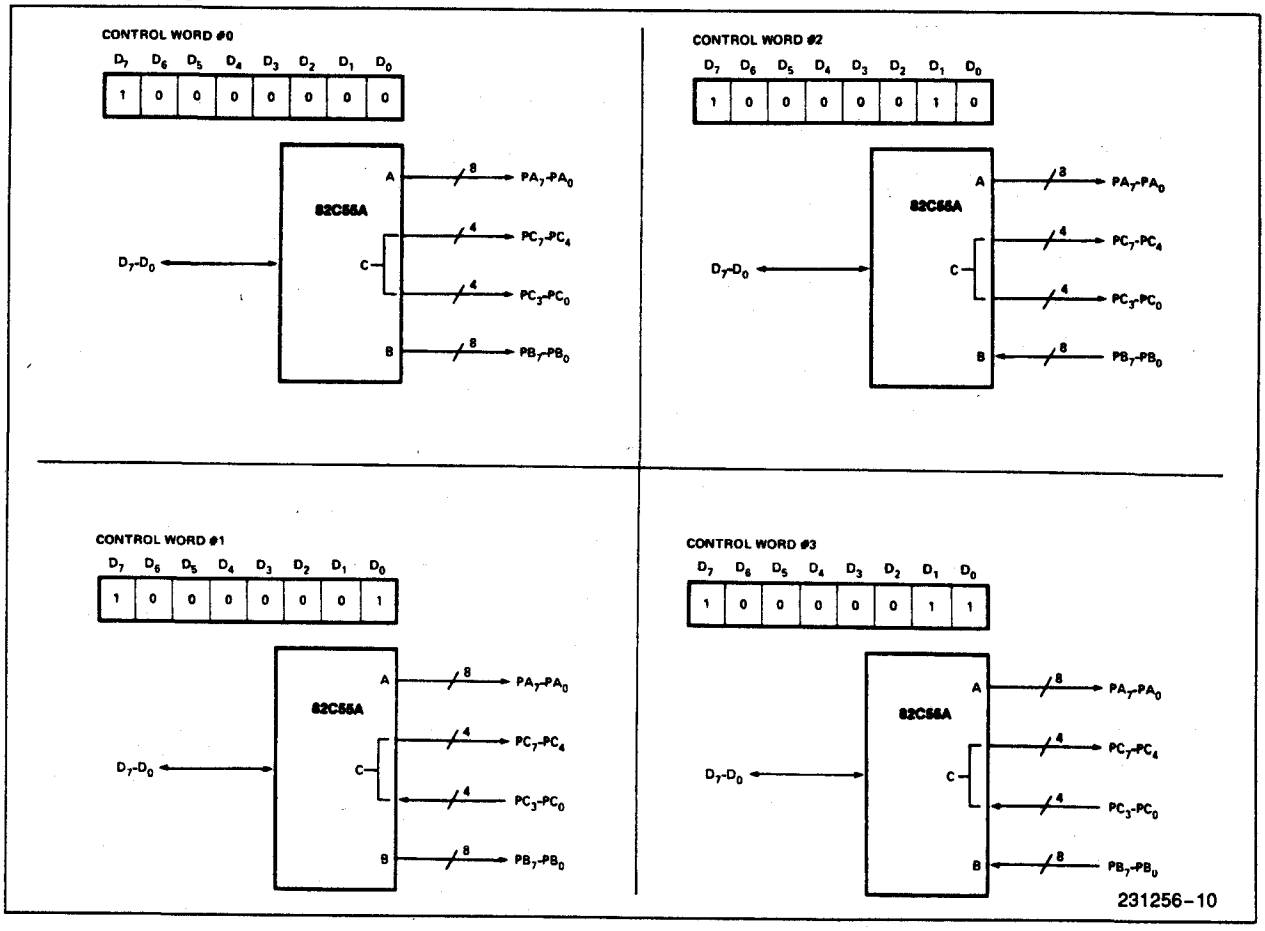

**MODE 0 (BASIC OUTPUT)**



## MODE 0 Port Definition

| A | | B | | GROUP A | | | GROUP B | |
|---|---|---|---|---|---|---|---|---|
| $D_4$ | $D_3$ | $D_1$ | $D_0$ | PORT A | PORT C (UPPER) | # | PORT B | PORT C (LOWER) |
| 0 | 0 | 0 | 0 | OUTPUT | OUTPUT | 0 | OUTPUT | OUTPUT |
| 0 | 0 | 0 | 1 | OUTPUT | OUTPUT | 1 | OUTPUT | INPUT |
| 0 | 0 | 1 | 0 | OUTPUT | OUTPUT | 2 | INPUT | OUTPUT |
| 0 | 0 | 1 | 1 | OUTPUT | OUTPUT | 3 | INPUT | INPUT |
| 0 | 1 | 0 | 0 | OUTPUT | INPUT | 4 | OUTPUT | OUTPUT |
| 0 | 1 | 0 | 1 | OUTPUT | INPUT | 5 | OUTPUT | INPUT |
| 0 | 1 | 1 | 0 | OUTPUT | INPUT | 6 | INPUT | OUTPUT |
| 0 | 1 | 1 | 1 | OUTPUT | INPUT | 7 | INPUT | INPUT |
| 1 | 0 | 0 | 0 | INPUT | OUTPUT | 8 | OUTPUT | OUTPUT |
| 1 | 0 | 0 | 1 | INPUT | OUTPUT | 9 | OUTPUT | INPUT |
| 1 | 0 | 1 | 0 | INPUT | OUTPUT | 10 | INPUT | OUTPUT |
| 1 | 0 | 1 | 1 | INPUT | OUTPUT | 11 | INPUT | INPUT |
| 1 | 1 | 0 | 0 | INPUT | INPUT | 12 | OUTPUT | OUTPUT |
| 1 | 1 | 0 | 1 | INPUT | INPUT | 13 | OUTPUT | INPUT |
| 1 | 1 | 1 | 0 | INPUT | INPUT | 14 | INPUT | OUTPUT |
| 1 | 1 | 1 | 1 | INPUT | INPUT | 15 | INPUT | INPUT |

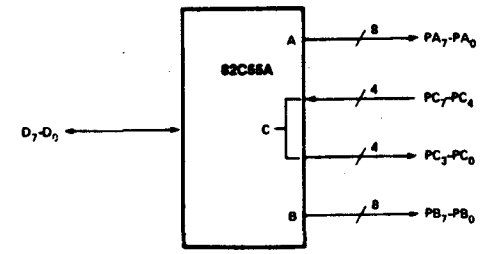
## MODE 0 Configurations



## MODE 0 Configurations (Continued)

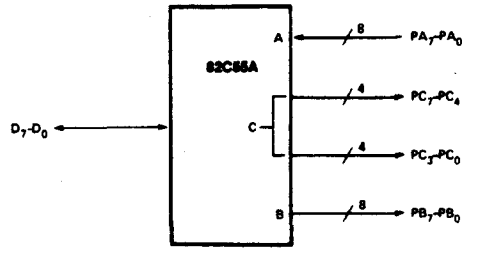
CONTROL WORD #4

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |



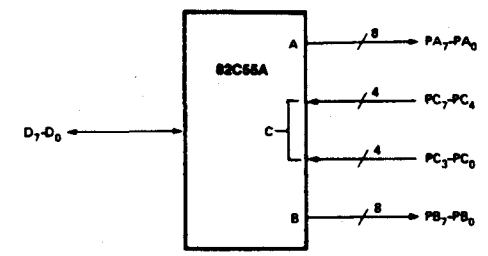

CONTROL WORD #8

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |



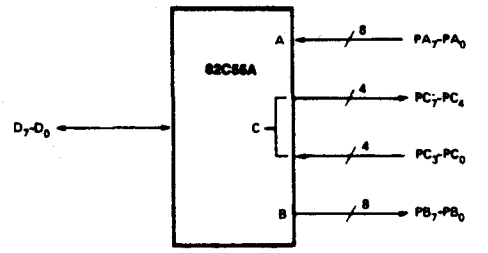

CONTROL WORD #5

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |



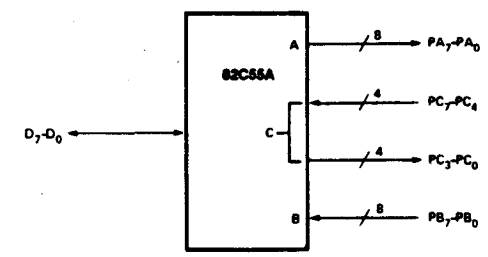

CONTROL WORD #9

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |



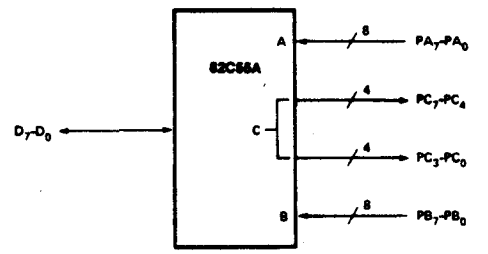

CONTROL WORD #6

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 |



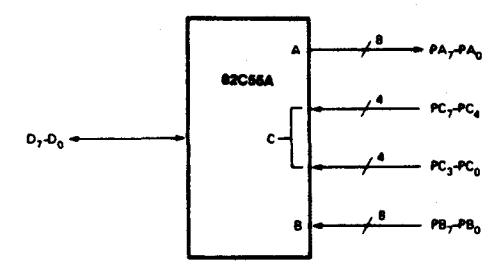

CONTROL WORD #10

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |



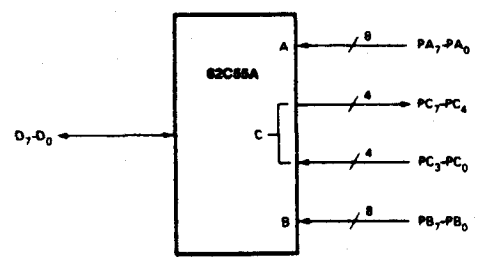

CONTROL WORD #7

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |



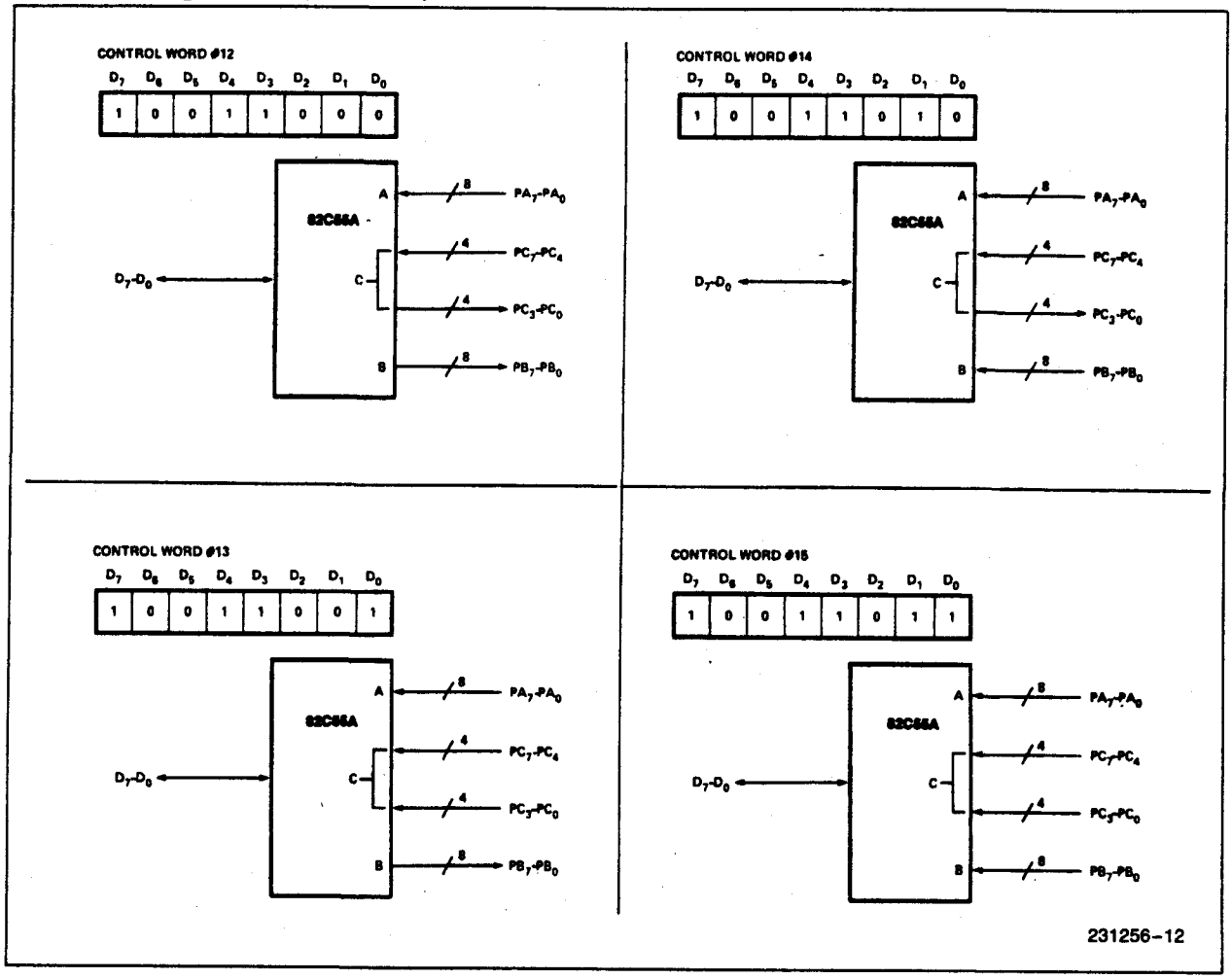

CONTROL WORD #11

| $D_7$ | $D_6$ | $D_5$ | $D_4$ | $D_3$ | $D_2$ | $D_1$ | $D_0$ |
|---|---|---|---|---|---|---|---|
| 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |




231256-11

**MODE 0 Configurations** (Continued)




## Operating Modes

**MODE 1 (Strobed Input/Output).** This functional configuration provides a means for transferring I/O data to or from a specified port in conjunction with strobes or "handshaking" signals. In mode 1, Port A and Port B use the lines on Port C to generate or accept these "handshaking" signals.

Mode 1 Basic functional Definitions:

- Two Groups (Group A and Group B).
- Each group contains one 8-bit data port and one 4-bit control/data port.
- The 8-bit data port can be either input or output Both inputs and outputs are latched.
- The 4-bit port is used for control and status of the 8-bit data port.

### Input Control Signal Definition

**$\overline{STB}$ (Strobe Input).** A "low" on this input loads data into the input latch.

### IBF (Input Buffer Full F/F)

A "high" on this output indicates that the data has been loaded into the input latch; in essence, an acknowledgement. IBF is set by $\overline{STB}$ input being low and is reset by the rising edge of the $\overline{RD}$ input.

### INTR (Interrupt Request)

A "high" on this output can be used to interrupt the CPU when an input device is requesting service. INTR is set by the $\overline{STB}$ is a "one", IBF is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{RD}$. This procedure allows an input device to request service from the CPU by simply strobing its data into the port.

**INTE A**

Controlled by bit set/reset of $PC_4$.

**INTE B**

Controlled by bit set/reset of $PC_2$.

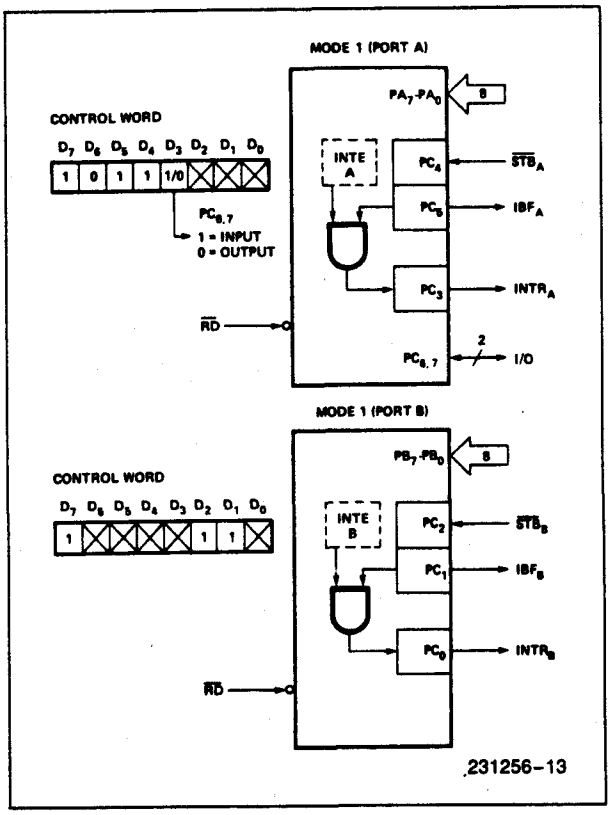



**Figure 8. MODE 1 Input**

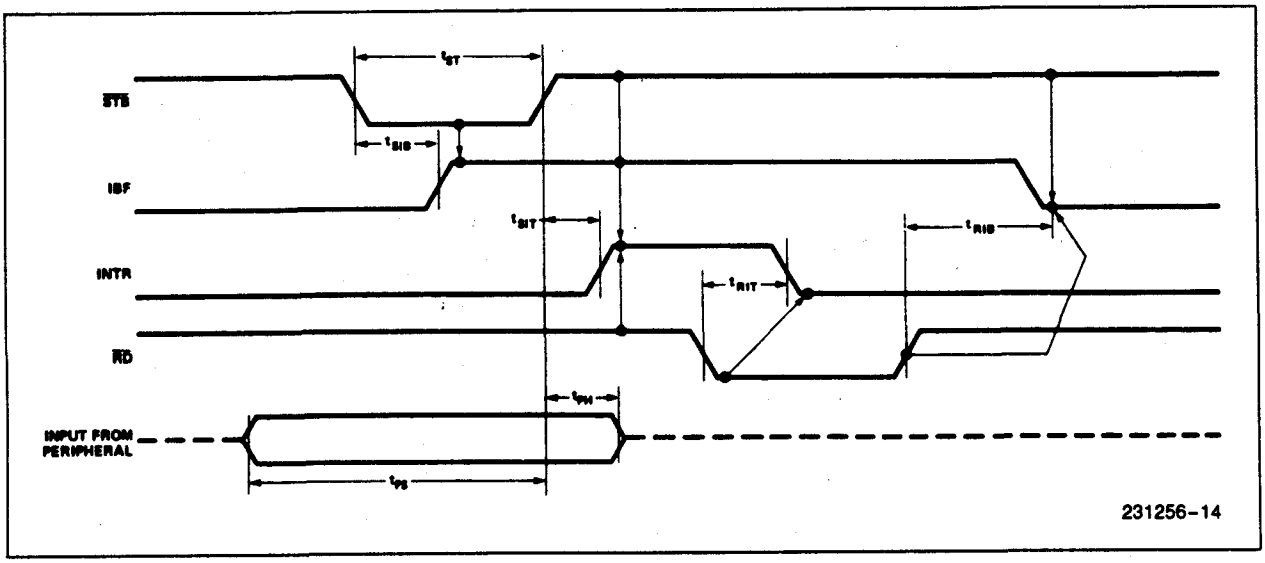



**Figure 9. MODE 1 (Strobed Input)**

### Output Control Signal Definition

**$\overline{OBF}$ (Output Buffer Full F/F).** The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to the specified port. The $\overline{OBF}$ F/F will be set by the rising edge of the $\overline{WR}$ input and reset by $\overline{ACK}$ Input being low.

**$\overline{ACK}$ (Acknowledge Input).** A "low" on this input informs the 82C55A that the data from Port A or Port B has been accepted. In essence, a response from the peripheral device indicating that it has received the data output by the CPU.

**INTR (Interrupt Request).** A "high" on this output can be used to interrupt the CPU when an output device has accepted data transmitted by the CPU. INTR is set when $\overline{ACK}$ is a "one", $\overline{OBF}$ is a "one" and INTE is a "one". It is reset by the falling edge of $\overline{WR}$.

**INTE A**

Controlled by bit set/reset of $PC_6$.

**INTE B**

Controlled by bit set/reset of $PC_2$.

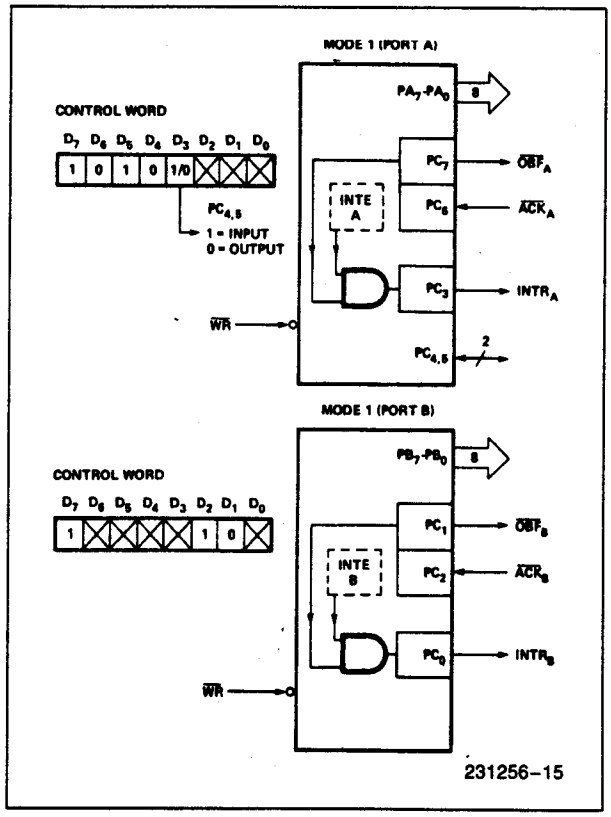



**Figure 10. MODE 1 Output**

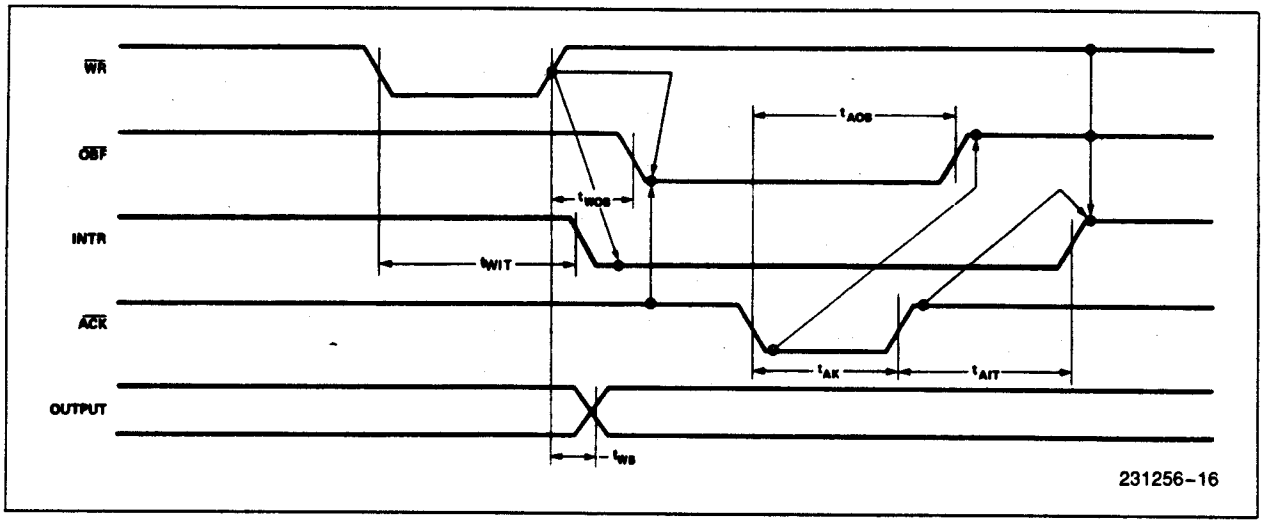



**Figure 11. MODE 1 (Strobed Output)**

### Combinations of MODE 1

Port A and Port B can be individually defined as input or output in Mode 1 to support a wide variety of strobed I/O applications.

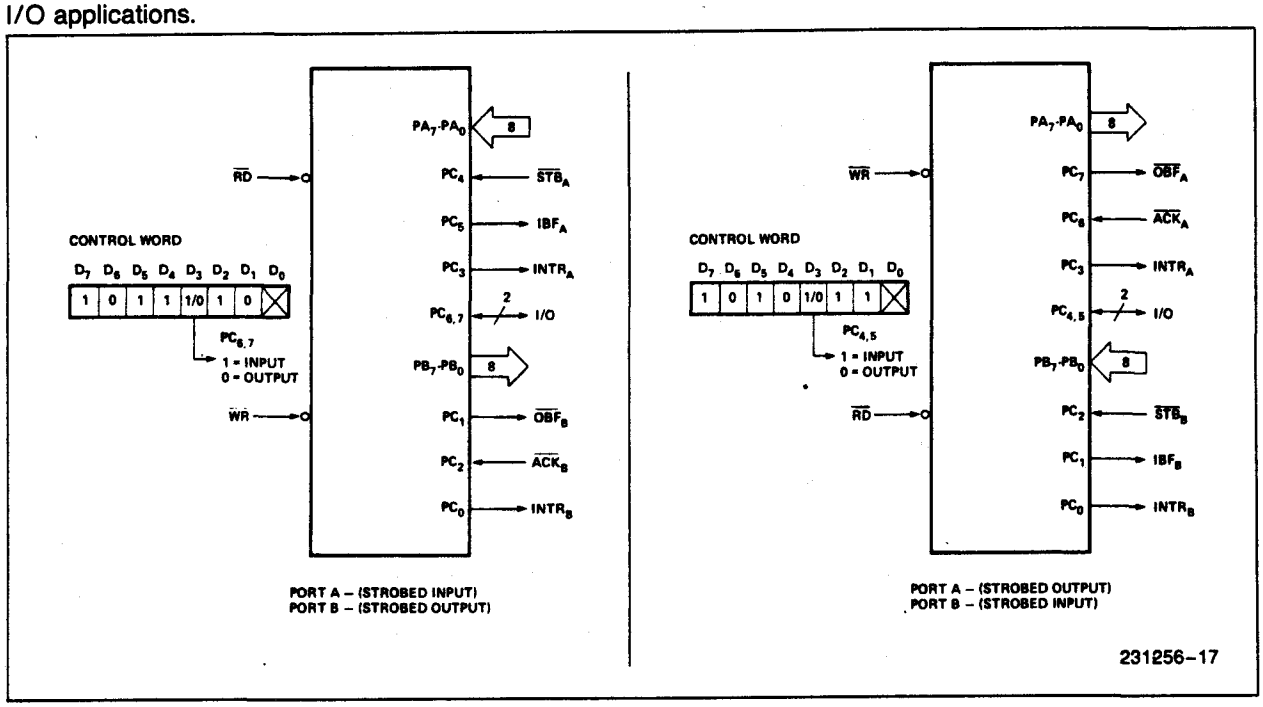



**Figure 12. Combinations of MODE 1**

### Operating Modes

**MODE 2 (Strobed Bidirectional Bus I/O).** This functional configuration provides a means for communicating with a peripheral device or structure on a single 8-bit bus for both transmitting and receiving data (bidirectional bus I/O). "Handshaking" signals are provided to maintain proper bus flow discipline in a similar manner to MODE 1. Interrupt generation and enable/disable functions are also available.

MODE 2 Basic Functional Definitions:

- Used in Group A **only**.
- One 8-bit, bi-directional bus port (Port A) and a 5-bit control port (Port C).
- Both inputs and outputs are latched.
- The 5-bit control port (Port C) is used for control and status for the 8-bit, bi-directional bus port (Port A).

### Bidirectional Bus I/O Control Signal Definition

**INTR (Interrupt Request).** A high on this output can be used to interrupt the CPU for input or output operations.

### Output Operations

**$\overline{OBF}$ (Output Buffer Full).** The $\overline{OBF}$ output will go "low" to indicate that the CPU has written data out to port A.

**$\overline{ACK}$ (Acknowledge).** A "low" on this input enables the tri-state output buffer of Port A to send out the data. Otherwise, the output buffer will be in the high impedance state.

**INTE 1 (The INTE Flip-Flop Associated with $\overline{OBF}$).** Controlled by bit set/reset of $PC_6$.

### Input Operations

**$\overline{STB}$ (Strobe Input).** A "low" on this input loads data into the input latch.

**IBF (Input Buffer Full F/F).** A "high" on this output indicates that data has been loaded into the input latch.

**INTE 2 (The INTE Flip-Flop Associated with IBF).** Controlled by bit set/reset of $PC_4$.

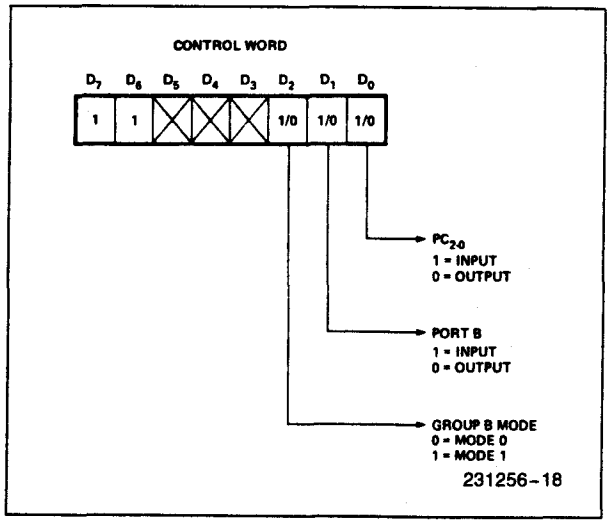



**Figure 13. MODE Control Word**

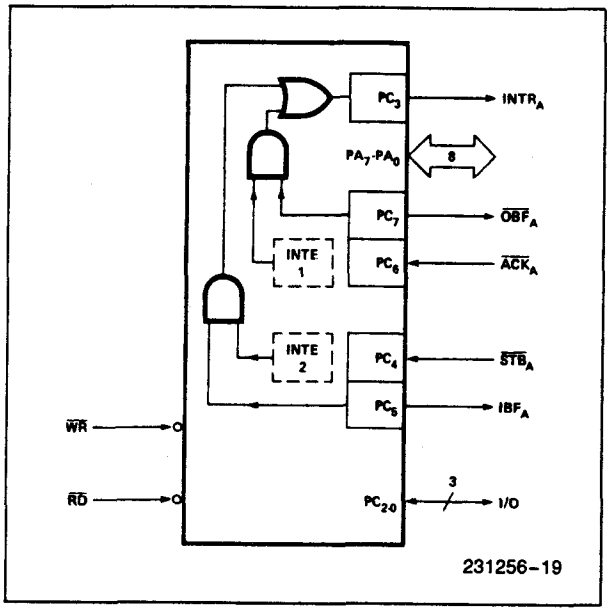



**Figure 14. MODE 2**

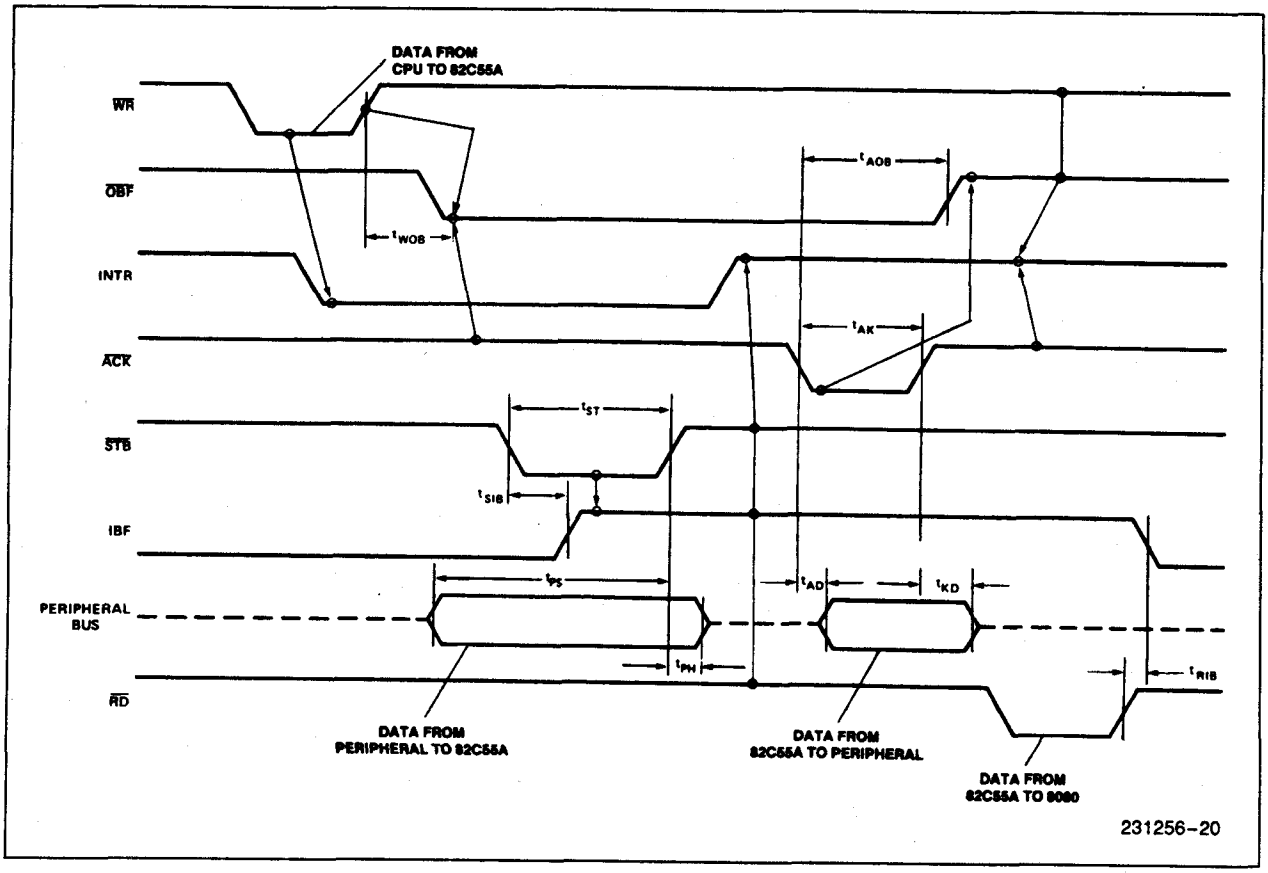



**Figure 15. MODE 2 (Bidirectional)**

**NOTE:**
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$, and $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
(INTR = IBF • $\overline{MASK}$ • $\overline{STB}$ • $\overline{RD}$ + $\overline{OBF}$ • $\overline{MASK}$ • $\overline{ACK}$ • $\overline{WR}$)

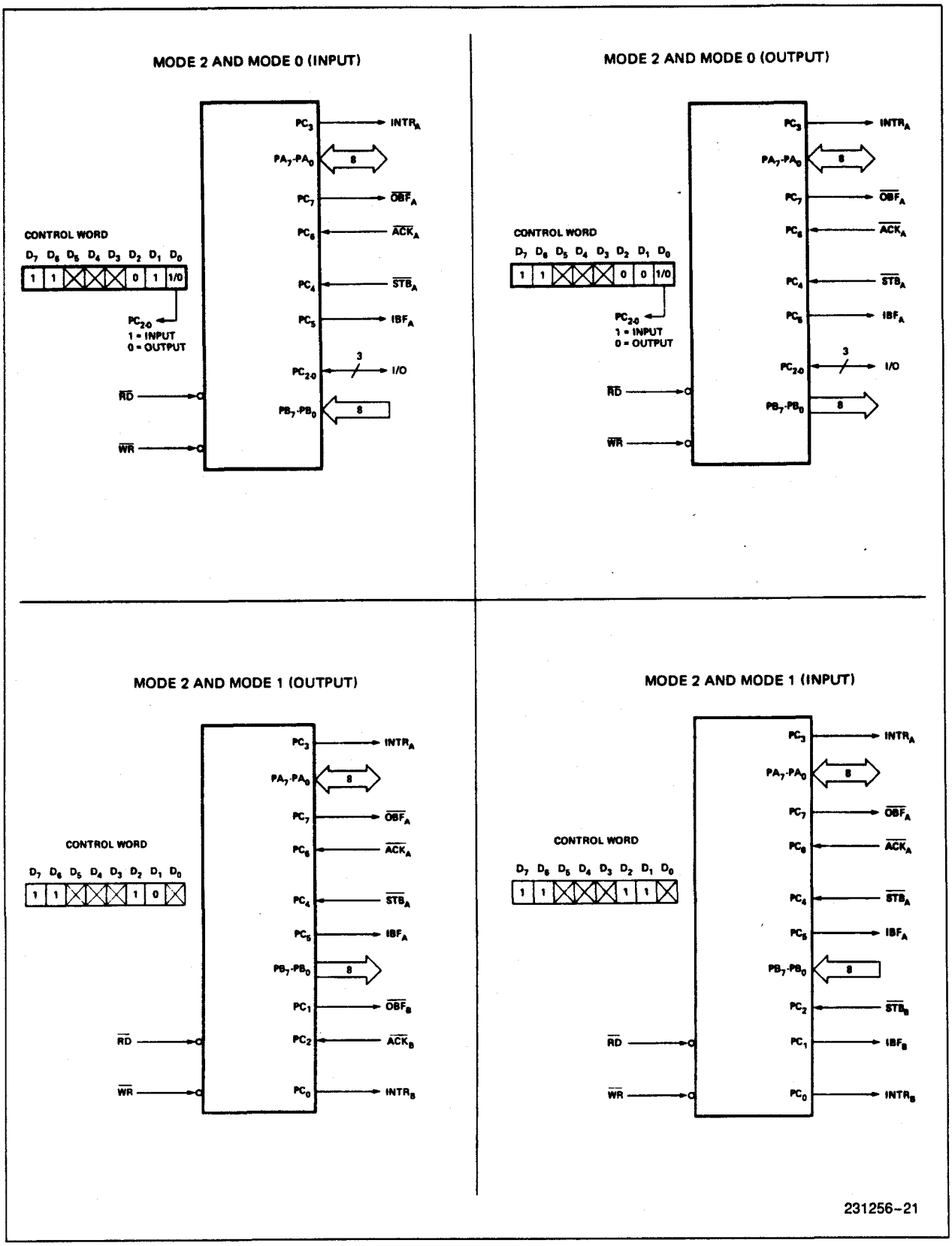



**Figure 16. MODE 1/4 Combinations**

**Mode Definition Summary**

| | MODE 0 | | MODE 1 | | MODE 2 | |
|---|---|---|---|---|---|---|
| | IN | OUT | IN | OUT | GROUP A ONLY | |
| $PA_0$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_1$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_2$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_3$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_4$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_5$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_6$ | IN | OUT | IN | OUT | ⟷ | |
| $PA_7$ | IN | OUT | IN | OUT | ⟷ | |
| $PB_0$ | IN | OUT | IN | OUT | — | MODE 0 OR MODE 1 ONLY |
| $PB_1$ | IN | OUT | IN | OUT | — | |
| $PB_2$ | IN | OUT | IN | OUT | — | |
| $PB_3$ | IN | OUT | IN | OUT | — | |
| $PB_4$ | IN | OUT | IN | OUT | — | |
| $PB_5$ | IN | OUT | IN | OUT | — | |
| $PB_6$ | IN | OUT | IN | OUT | — | |
| $PB_7$ | IN | OUT | IN | OUT | — | |
| $PC_0$ | IN | OUT | $INTR_B$ | $INTR_B$ | I/O | |
| $PC_1$ | IN | OUT | $IBF_B$ | $\overline{OBF}_B$ | I/O | |
| $PC_2$ | IN | OUT | $\overline{STB}_B$ | $\overline{ACK}_B$ | I/O | |
| $PC_3$ | IN | OUT | $INTR_A$ | $INTR_A$ | $INTR_A$ | |
| $PC_4$ | IN | OUT | $\overline{STB}_A$ | I/O | $\overline{STB}_A$ | |
| $PC_5$ | IN | OUT | $IBF_A$ | I/O | $IBF_A$ | |
| $PC_6$ | IN | OUT | I/O | $\overline{ACK}_A$ | $\overline{ACK}_A$ | |
| $PC_7$ | IN | OUT | I/O | $\overline{OBF}_A$ | $\overline{OBF}_A$ | |

**Special Mode Combination Considerations**

There are several combinations of modes possible. For any combination, some or all of the Port C lines are used for control or status. The remaining bits are either inputs or outputs as defined by a "Set Mode" command.

During a read of Port C, the state of all the Port C lines, except the $\overline{ACK}$ and $\overline{STB}$ lines, will be placed on the data bus. In place of the $\overline{ACK}$ and $\overline{STB}$ line states, flag status will appear on the data bus in the PC2, PC4, and PC6 bit positions as illustrated by Figure 18.

Through a "Write Port C" command, only the Port C pins programmed as outputs in a Mode 0 group can be written. No other pins can be affected by a "Write Port C" command, nor can the interrupt enable flags be accessed. To write to any Port C output programmed as an output in a Mode 1 group or to change an interrupt enable flag, the "Set/Reset Port C Bit" command must be used.

With a "Set/Reset Port C Bit" command, any Port C line programmed as an output (including INTR, IBF and $\overline{OBF}$) can be written, or an interrupt enable flag can be either set or reset. Port C lines programmed as inputs, including $\overline{ACK}$ and $\overline{STB}$ lines, associated with Port C are not affected by a "Set/Reset Port C Bit" command. Writing to the corresponding Port C bit positions of the $\overline{ACK}$ and $\overline{STB}$ lines with the "Set/Reset Port C Bit" command will affect the Group A and Group B interrupt enable flags, as illustrated in Figure 18.

**Current Drive Capability**

Any output on Port A, B or C can sink or source 2.5 mA. This feature allows the 82C55A to directly drive Darlington type drivers and high-voltage displays that require such sink or source current.

### Reading Port C Status

In Mode 0, Port C transfers data to or from the peripheral device. When the 82C55A is programmed to function in Modes 1 or 2, Port C generates or accepts "hand-shaking" signals with the peripheral device. Reading the contents of Port C allows the programmer to test or verify the "status" of each peripheral device and change the program flow accordingly.

There is no special instruction to read the status information from Port C. A normal read operation of Port C is executed to perform this function.

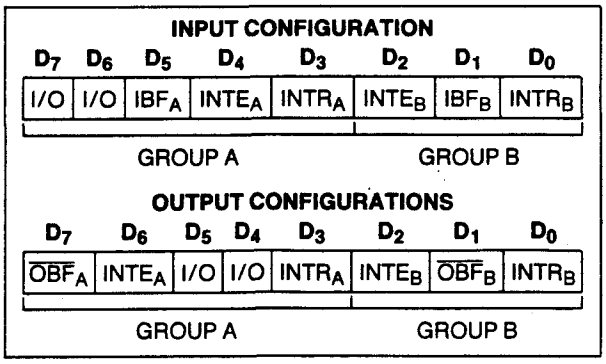



**Figure 17a. MODE 1 Status Word Format**

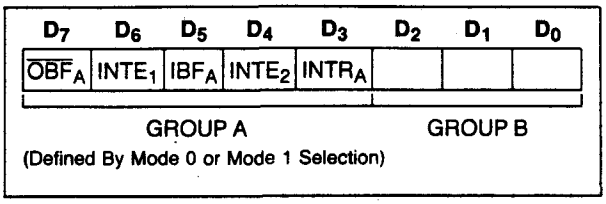



**Figure 17b. MODE 2 Status Word Format**

| Interrupt Enable Flag | Position | Alternate Port C Pin Signal (Mode) |
|---|---|---|
| INTE B | PC2 | $\overline{ACK}_B$ (Output Mode 1) or $\overline{STB}_B$ (Input Mode 1) |
| INTE A2 | PC4 | $\overline{STB}_A$ (Input Mode 1 or Mode 2) |
| INTE A1 | PC6 | $\overline{ACK}_A$ (Output Mode 1 or Mode 2 |

**Figure 18. Interrupt Enable Flags in Modes 1 and 2**

## ABSOLUTE MAXIMUM RATINGS*

Ambient Temperature Under Bias....0°C to + 70°C
Storage Temperature .....− 65°C to + 150°C
Supply Voltage .....− 0.5 to + 8.0V
Operating Voltage .....+ 4V to + 7V
Voltage on any Input.....GND−2V to + 6.5V
Voltage on any Output ..GND−0.5V to $V_{CC}$ + 0.5V
Power Dissipation .....1 Watt

**Notice: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.*

## D.C. CHARACTERISTICS

$T_A$ = 0°C to 70°C, $V_{CC}$ = +5V ±10%, GND = 0V ($T_A$ = −40°C to +85°C for Extended Temperture)

| Symbol | Parameter | Min | Max | Units | Test Conditions |
|---|---|---|---|---|---|
| $V_{IL}$ | Input Low Voltage | −0.5 | 0.8 | V | |
| $V_{IH}$ | Input High Voltage | 2.0 | $V_{CC}$ | V | |
| $V_{OL}$ | Output Low Voltage | | 0.4 | V | $I_{OL}$ = 2.5 mA |
| $V_{OH}$ | Output High Voltage | 3.0<br>$V_{CC}$ − 0.4 | | V<br>V | $I_{OH}$ = −2.5 mA<br>$I_{OH}$ = −100 μA |
| $I_{IL}$ | Input Leakage Current | | ±1 | μA | $V_{IN}$ = $V_{CC}$ to 0V (Note 1) |
| $I_{OFL}$ | Output Float Leakage Current | | ±10 | μA | $V_{IN}$ = $V_{CC}$ to 0V (Note 2) |
| $I_{DAR}$ | Darlington Drive Current | ±2.5 | (Note 4) | mA | Ports A, B, C<br>$R_{ext}$ = 500Ω<br>$V_{ext}$ = 1.7V |
| $I_{PHL}$ | Port Hold Low Leakage Current | +50 | +300 | μA | $V_{OUT}$ = 1.0V<br>Port A only |
| $I_{PHH}$ | Port Hold High Leakage Current | −50 | −300 | μA | $V_{OUT}$ = 3.0V<br>Ports A, B, C |
| $I_{PHLO}$ | Port Hold Low Overdrive Current | −350 | | μA | $V_{OUT}$ = 0.8V |
| $I_{PHHO}$ | Port Hold High Overdrive Current | +350 | | μA | $V_{OUT}$ = 3.0V |
| $I_{CC}$ | $V_{CC}$ Supply Current | | 10 | mA | (Note 3) |
| $I_{CCSB}$ | $V_{CC}$ Supply Current-Standby | | 10 | μA | $V_{CC}$ = 5.5V<br>$V_{IN}$ = $V_{CC}$ or GND<br>Port Conditions<br>If I/P = Open/High<br>O/P = Open Only<br>With Data Bus =<br>High/Low<br>$\overline{CS}$ = High<br>Reset = Low<br>Pure Inputs =<br>Low/High |

**NOTES:**
1. Pins $A_1$, $A_0$, $\overline{CS}$, $\overline{WR}$, $\overline{RD}$, Reset.
2. Data Bus; Ports B, C.
3. Outputs open.
4. Limit output current to 4.0 mA.

## CAPACITANCE

$T_A$ = 25°C, $V_{CC}$ = GND = 0V

| Symbol | Parameter | Min | Max | Units | Test Conditions |
|---|---|---|---|---|---|
| $C_{IN}$ | Input Capacitance | | 10 | pF | Unmeasured pins returned to GND $f_c$ = 1 MHz(5) |
| $C_{I/O}$ | I/O Capacitance | | 20 | pF | |

**NOTE:**
5. Sampled not 100% tested.

## A.C. CHARACTERISTICS

$T_A$ = 0° to 70°C, $V_{CC}$ = +5V ±10%, GND = 0V

$T_A$ = −40°C to +85°C for Extended Temperature

### BUS PARAMETERS

### READ CYCLE

| Symbol | Parameter | 82C55A-2 | | Units | Test Conditions |
|---|---|---|---|---|---|
| | | Min | Max | | |
| $t_{AR}$ | Address Stable Before $\overline{RD}\downarrow$ | 0 | | ns | |
| $t_{RA}$ | Address Hold Time After $\overline{RD}\uparrow$ | 0 | | ns | |
| $t_{RR}$ | $\overline{RD}$ Pulse Width | 150 | | ns | |
| $t_{RD}$ | Data Delay from $\overline{RD}\downarrow$ | | 120 | ns | |
| $t_{DF}$ | $\overline{RD}\uparrow$ to Data Floating | 10 | 75 | ns | |
| $t_{RV}$ | Recovery Time between $\overline{RD}/\overline{WR}$ | 200 | | ns | |

### WRITE CYCLE

| Symbol | Parameter | 82C55A-2 | | Units | Test Conditions |
|---|---|---|---|---|---|
| | | Min | Max | | |
| $t_{AW}$ | Address Stable Before $\overline{WR}\downarrow$ | 0 | | ns | |
| $t_{WA}$ | Address Hold Time After $\overline{WR}\uparrow$ | 20 | | ns | Ports A & B |
| | | 20 | | ns | Port C |
| $t_{WW}$ | $\overline{WR}$ Pulse Width | 100 | | ns | |
| $t_{DW}$ | Data Setup Time Before $\overline{WR}\uparrow$ | 100 | | ns | |
| $t_{WD}$ | Data Hold Time After $\overline{WR}\uparrow$ | 30 | | ns | Ports A & B |
| | | 30 | | ns | Port C |

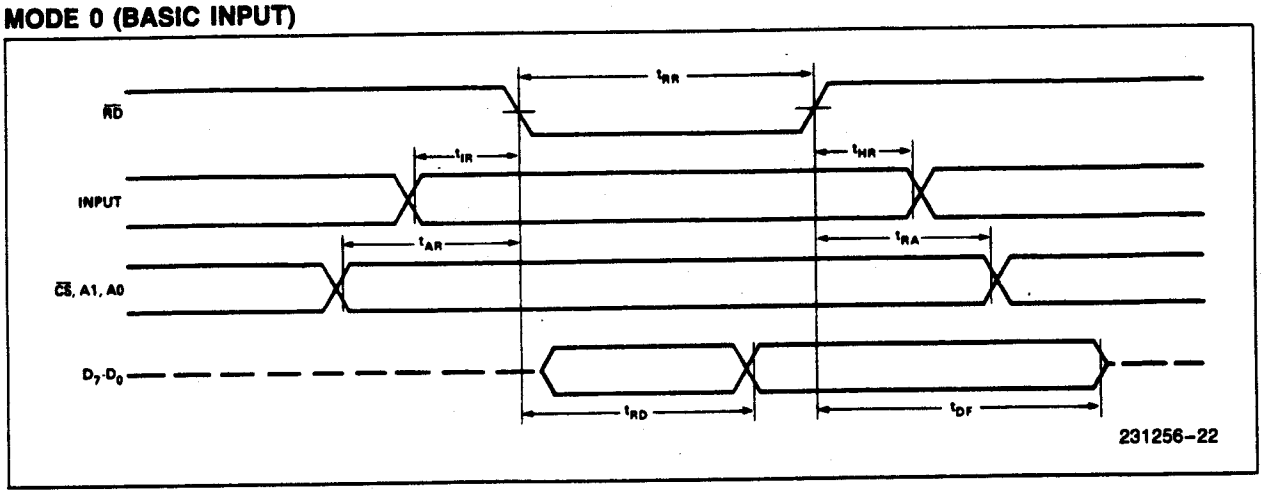
## OTHER TIMINGS

| Symbol | Parameter | 82C55A-2 | | Units Conditions | Test |
|---|---|---|---|---|---|
| | | Min | Max | | |
| $t_{WB}$ | $\overline{WR}$ = 1 to Output | | 350 | ns | |
| $t_{IR}$ | Peripheral Data Before $\overline{RD}$ | 0 | | ns | |
| $t_{HR}$ | Peripheral Data After $\overline{RD}$ | 0 | | ns | |
| $t_{AK}$ | $\overline{ACK}$ Pulse Width | 200 | | ns | |
| $t_{ST}$ | $\overline{STB}$ Pulse Width | 100 | | ns | |
| $t_{PS}$ | Per. Data Before $\overline{STB}$ High | 20 | | ns | |
| $t_{PH}$ | Per. Data After $\overline{STB}$High | 50 | | ns | |
| $t_{AD}$ | $\overline{ACK}$ = 0 to Output | | 175 | ns | |
| $t_{KD}$ | $\overline{ACK}$ = 1 to Output Float | 20 | 250 | ns | |
| $t_{WOB}$ | $\overline{WR}$ = 1 to $\overline{OBF}$ = 0 | | 150 | ns | |
| $t_{AOB}$ | $\overline{ACK}$ = 0 to $\overline{OBF}$ = 1 | | 150 | ns | |
| $t_{SIB}$ | $\overline{STB}$ = 0 to IBF = 1 | | 150 | ns | |
| $t_{RIB}$ | $\overline{RD}$ = 1 to IBF = 0 | | 150 | ns | |
| $t_{RIT}$ | $\overline{RD}$ = 0 to INTR = 0 | | 200 | ns | |
| $t_{SIT}$ | $\overline{STB}$ = 1 to INTR = 1 | | 150 | ns | |
| $t_{AIT}$ | $\overline{ACK}$ = 1 to INTR = 1 | | 150 | ns | |
| $t_{WIT}$ | $\overline{WR}$ = 0 to INTR = 0 | | 200 | ns | see note 1 |
| $t_{RES}$ | Reset Pulse Width | 500 | | ns | see note 2 |

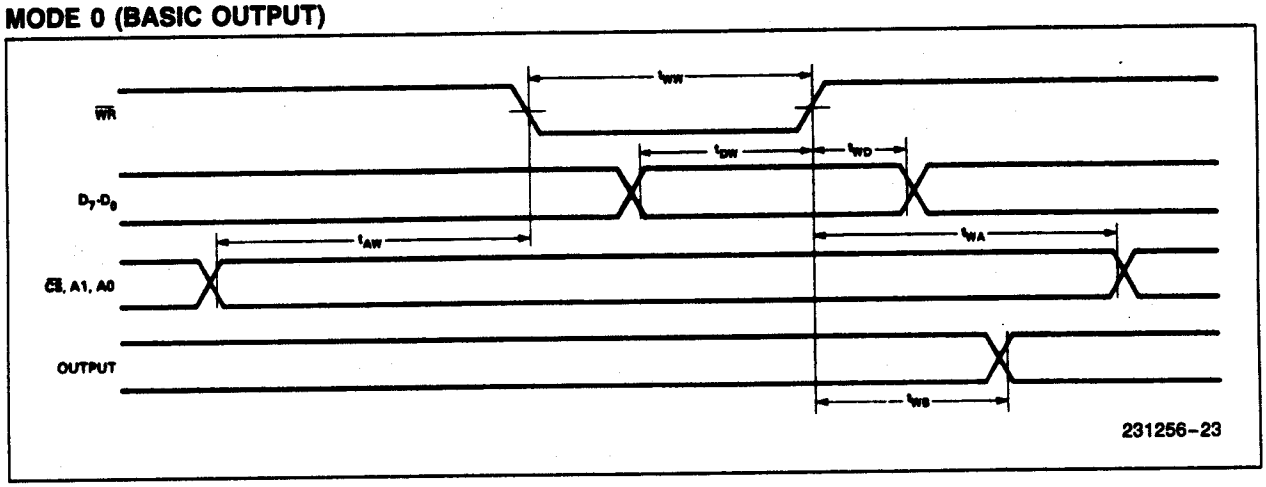
**NOTE:**
1. INTR ↑ may occur as early as $\overline{WR}$ ↓.
2. Pulse width of initial Reset pulse after power on must be at least 50 μSec. Subsequent Reset pulses may be 500 ns minimum.

## WAVEFORMS

### MODE 0 (BASIC INPUT)

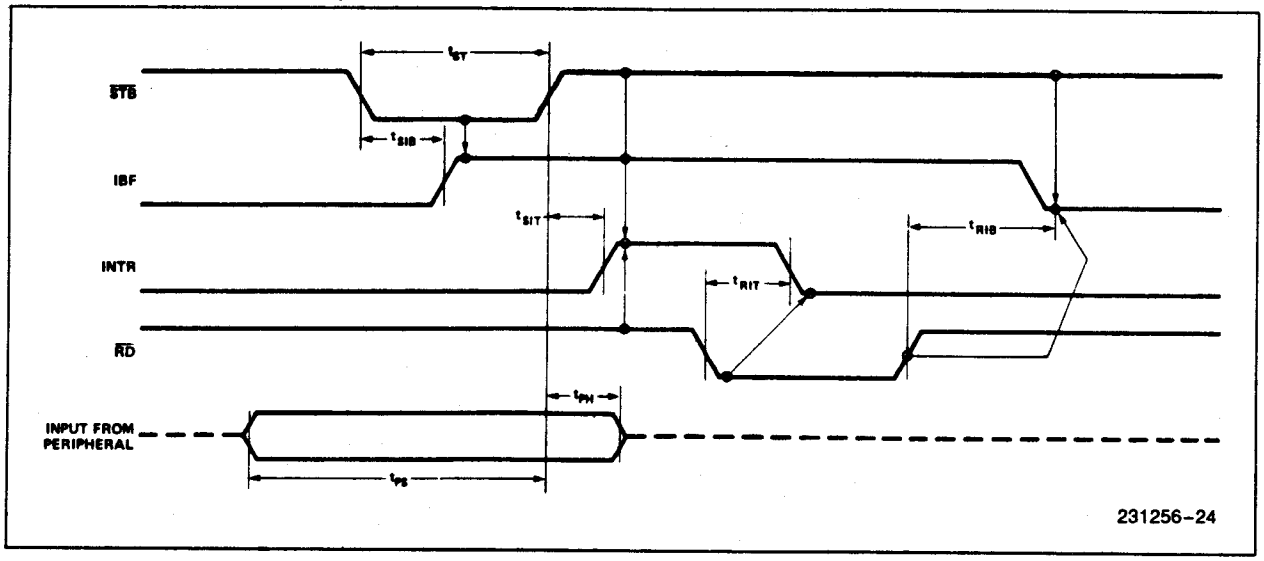



### MODE 0 (BASIC OUTPUT)

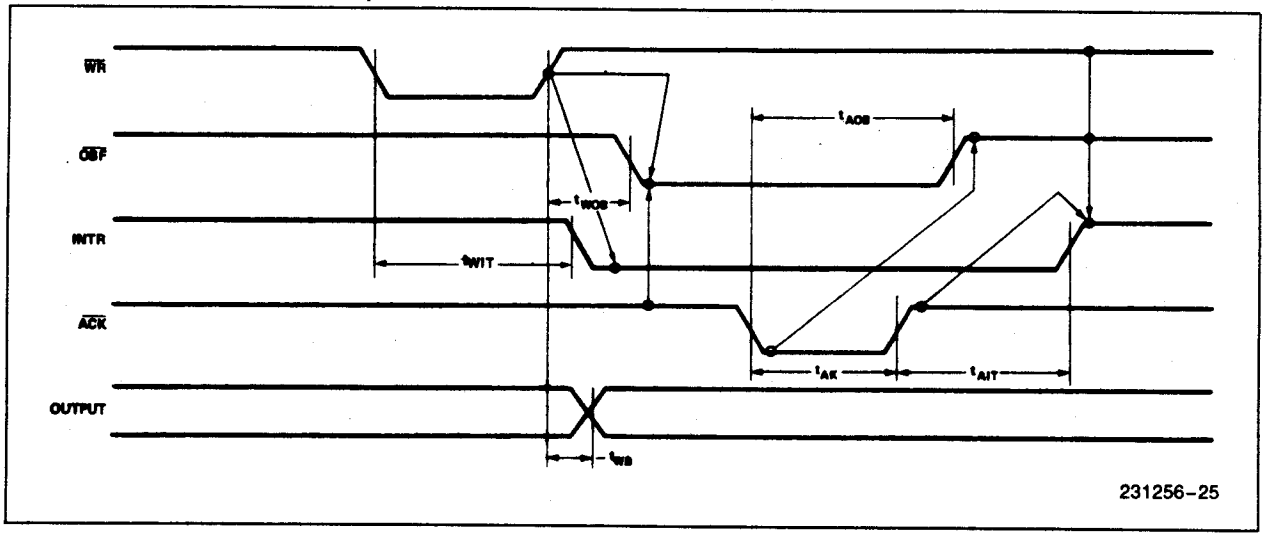


## WAVEFORMS (Continued)

### MODE 1 (STROBED INPUT)

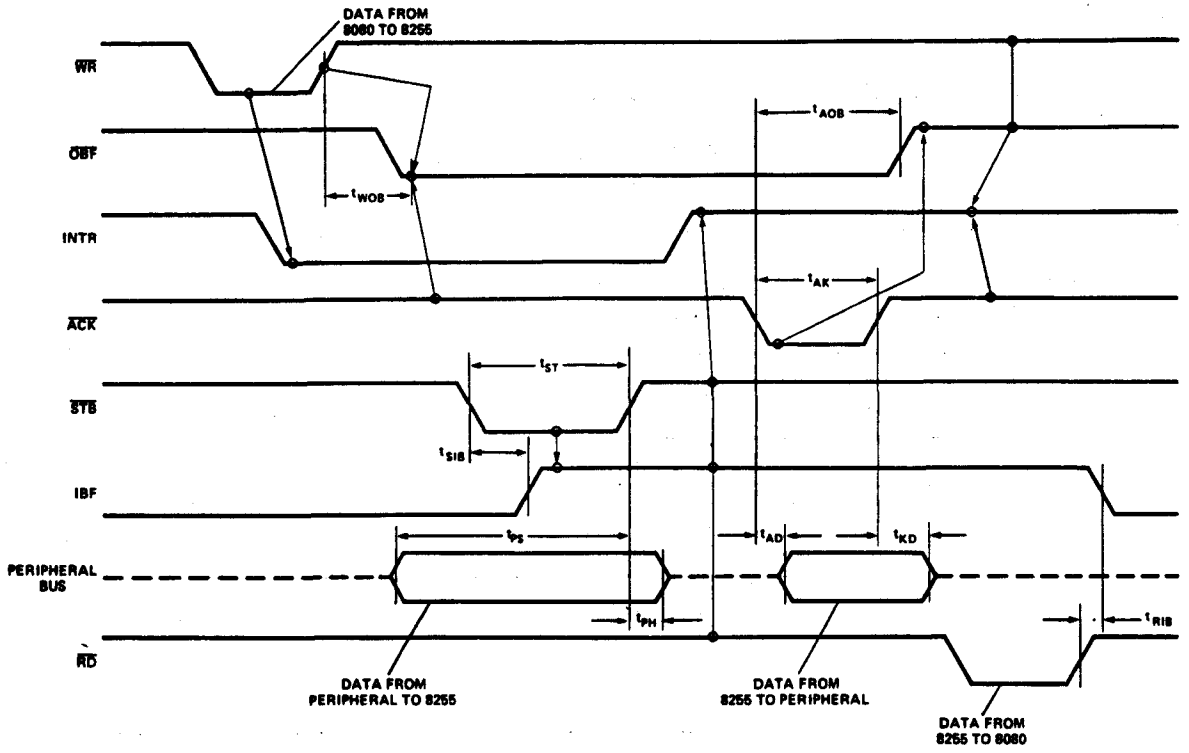



### MODE 1 (STROBED OUTPUT)



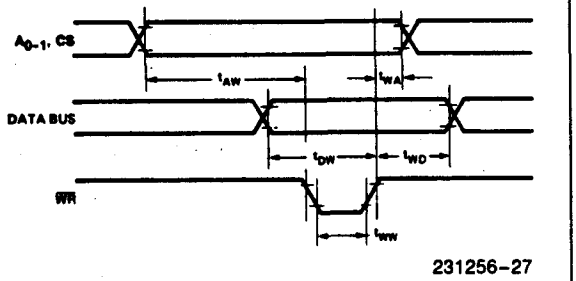
## WAVEFORMS (Continued)

### MODE 2 (BIDIRECTIONAL)

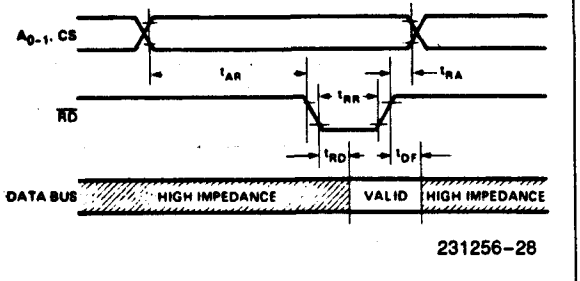



**Note:**
Any sequence where $\overline{WR}$ occurs before $\overline{ACK}$ AND $\overline{STB}$ occurs before $\overline{RD}$ is permissible.
($INTR = IBF \bullet \overline{MASK} \bullet \overline{STB} \bullet \overline{RD} + \overline{OBF} \bullet \overline{MASK} \bullet \overline{ACK} \bullet \overline{WR}$)

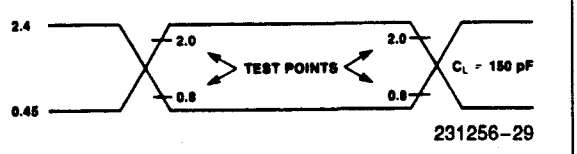
### WRITE TIMING




### READ TIMING

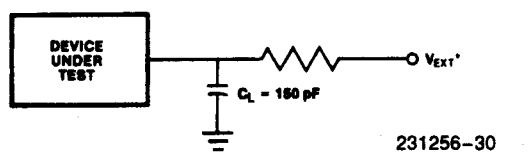



### A.C. TESTING INPUT, OUTPUT WAVEFORM




A.C. Testing Inputs Are Driven At 2.4V For A Logic 1 And 0.45V For A Logic 0 Timing Measurements Are Made At 2.0V For A Logic 1 And 0.8 For A Logic 0.

### A.C. TESTING LOAD CIRCUIT




*$V_{EXT}$ Is Set At Various Voltages During Testing To Guarantee The Specification. $C_L$ Includes Jig Capacitance.

# APPENDIX D

## WARRANTY

# LIMITED WARRANTY

Real Time Devices, Inc. warrants the hardware and software products it manufactures and produces to be free from defects in materials and workmanship for one year following the date of shipment from REAL TIME DEVICES. This warranty is limited to the original purchaser of product and is not transferable.

During the one year warranty period, REAL TIME DEVICES will repair or replace, at its option, any defective products or parts at no additional charge, provided that the product is returned, shipping prepaid, to REAL TIME DEVICES. All replaced parts and products become the property of REAL TIME DEVICES. **Before returning any product for repair, customers are required to contact the factory for an RMA number.**

THIS LIMITED WARRANTY DOES NOT EXTEND TO ANY PRODUCTS WHICH HAVE BEEN DAMAGED AS A RESULT OF ACCIDENT, MISUSE, ABUSE (such as: use of incorrect input voltages, improper or insufficient ventilation, failure to follow the operating instructions that are provided by REAL TIME DEVICES, "acts of God" or other contingencies beyond the control of REAL TIME DEVICES), OR AS A RESULT OF SERVICE OR MODIFICATION BY ANYONE OTHER THAN REAL TIME DEVICES. EXCEPT AS EXPRESSLY SET FORTH ABOVE, NO OTHER WARRANTIES ARE EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, AND REAL TIME DEVICES EXPRESSLY DISCLAIMS ALL WARRANTIES NOT STATED HEREIN. ALL IMPLIED WARRANTIES, INCLUDING IMPLIED WARRANTIES FOR MECHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, ARE LIMITED TO THE DURATION OF THIS WARRANTY. IN THE EVENT THE PRODUCT IS NOT FREE FROM DEFECTS AS WARRANTED ABOVE, THE PURCHASER'S SOLE REMEDY SHALL BE REPAIR OR REPLACEMENT AS PROVIDED ABOVE. UNDER NO CIRCUMSTANCES WILL REAL TIME DEVICES BE LIABLE TO THE PURCHASER OR ANY USER FOR ANY DAMAGES, INCLUDING ANY INCIDENTAL OR CONSEQUENTIAL DAMAGES, EXPENSES, LOST PROFITS, LOST SAVINGS, OR OTHER DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PRODUCT.

SOME STATES DO NOT ALLOW THE EXCLUSION OR LIMITATION OF INCIDENTAL OR CONSEQUENTIAL DAMAGES FOR CONSUMER PRODUCTS, AND SOME STATES DO NOT ALLOW LIMITATIONS ON HOW LONG AN IMPLIED WARRANTY LASTS, SO THE ABOVE LIMITATIONS OR EXCLUSIONS MAY NOT APPLY TO YOU.

THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS, AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

| TC1024 Board User-Selected Settings | |
|---|---|
| **Base I/O Address:** | |
| (hex) | (decimal) |
| **Interrupts:** | |
| Source: | IRQ Channel: |
| Source: | IRQ Channel: |
| Source: | IRQ Channel: |
| Source: | IRQ Channel: |
| Source: | IRQ Channel: |
| Source: | IRQ Channel: |